Infopark CMS Fiona

Portal Manager

Infopark CMS Fiona

Portal Manager

Inhalt

1 Vorbemerkungen

Dieses Dokument wendet sich an Administratoren, die den zu Infopark CMS Fiona gehörenden Portal Manager einsetzen möchten, um Portal-Funktionen wie Zugriffssteuerung, Channels, Personalisierung oder JSR-168-kompatible Portlets zu nutzen.

Das Dokument erläutert, wie der Portal Manager konfiguriert und in die gegebene Systemlandschaft integriert wird und wie existierende Portlets verwendet werden.

Die Verwendung sämtlicher mitgelieferter Portlets bis auf das Anmelde-Portlet (login portlet) ist lizenzierungspflichtig.

1.1 Systemvoraussetzungen

Die allgemeinen Systemvoraussetzungen gelten auch für den Portal Manager. Zusätzlich gilt:

- Wenn Sie Authentifizierung und Autorisierung über LDAP nutzen möchten, so benötigen Sie einen LDAP-Server.

2

2 Das Konzept des Portal Managers

2.1 Aufgaben des Portal Managers

Der zu Infopark CMS Fiona gehörende Portal Manager gibt Unternehmen die Möglichkeit, ihren Mitarbeitern oder Kunden verschiedenste Inhalte auf einer Oberfläche mit einheitlichem Look-and-Feel, d.h. ohne Brüche in Layout und Bedienung zu präsentieren.

- Die Inhalte lassen sich mit einem Zugriffsschutz versehen, so dass nur dazu berechtigte Personen nach der Anmeldung darauf zugreifen können.
- Ferner lassen sich die Inhalte dynamisieren, d.h. auf individuelle Benutzer oder Benutzergruppen zuschneiden.
- Nicht zuletzt können mit dem Portal Manager interaktive Komponenten mit Hilfe der Java-Portlet-Technologie (so genannte Portlets) in die Inhalte eingebunden werden.

Diese Funktionen können auf einfache Weise von Redakteuren oder Designern im Redaktionssystem verwendet werden.

2.1.1 Zugriffsschutz für Inhalte

Der Zugriffsschutz gibt Ihnen die Möglichkeit, Inhalte nur bestimmten Benutzern zugänglich zu machen, die Mitglied in einer bestimmten Benutzergruppe sind. Um dies bei einzelnen Benutzern festzustellen, müssen sich die Benutzer am Portal anmelden, sobald sie geschützte Inhalte anfragen. Wurden die Inhalte von einem nicht berechtigten Benutzer angefordert, so wird die Anfrage abgelehnt. Dies gilt sowohl für geschützte HTML-Seiten als auch für binäre Inhalte wie PDF-Dateien o.ä.

Zusätzlich ist es in vielen Fällen wünschenswert, dass Verweise auf solche Inhalte in anderen HTML-Seiten gar nicht oder abgewandelt (z.B. als Teaser) erscheinen, wenn nicht berechtigte Benutzer diese anderen Seiten ansehen. Der Portal Manager bietet eine Reihe von Funktionen, mit denen dies in unterschiedlichen Ausprägungen erreicht werden kann.

Mit dem Portal Manager können nicht nur Webseiten insgesamt mit einem Zugriffsschutz versehen werden, sondern auch Teile davon. Wenn Sie beispielsweise Ihren angemeldeten Nutzern besondere Dienste zur Verfügung stellen wollen, so können Sie die Links auf die entsprechenden Webseiten nur bei diesen Nutzern anzeigen lassen.

2.1.2 Dynamische Anpassung von Seiteninhalten

Oft soll das Aussehen einer Webseite erst im Moment der Auslieferung festgelegt werden. So lassen sich die Seiteninhalte beispielsweise anpassen, wenn der Benutzer angemeldet ist. Auch lassen sich Daten aus anderen Quellen "in letzter Sekunde" einfügen.

Ferner können Inhalte, die sich häufig ändern und zusätzlich an vielen Stellen auf der Website angezeigt werden müssen, dynamisch, d.h. bei der Auslieferung, einbinden. Dadurch brauchen diese Inhalte nur einmal erstellt zu werden. Dies kann den Export der Seiten, in die diese Inhalte eingebettet werden, drastisch beschleunigen.

2.1.3 Portlets

Portlets sind interaktive, eigenständige Teilkomponenten einer HTML-Seite. Typische Anwendungsfälle für Portlets sind beispielsweise Bewertung (*Voting*), Anmeldung (*Login*), mehrseitige Eingabe-Formulare oder News-Ticker.

Der Portal Manager enthält einen JSR-168-kompatiblen Portlet-Container und unterstützt so die zukunftssichere und modularisierte Entwicklung von interaktiven Komponenten. Andere Portale erfordern typischerweise eine mehr oder minder feste Anordnung von Portlets, in denen der Content dargestellt wird (kurz "Content in Portlets"). Diese Anordnung wird meist bei der Konfiguration des Portals festgelegt und ist später nur mit großem Aufwand und von Fachleuten änderbar.

Der Portal Manager ermöglicht es hingegen, Portlets auf beliebigen Seiten zu platzieren, die mit dem CMS erstellt und gepflegt werden, und zwar in beliebiger Anzahl und Anordnung (kurz: „Portlets im Content“).

Gegenüber anderen Mechanismen wie PHP haben Java-basierte Portlets unter anderem die folgenden Vorteile:

- Der Code ist von den Inhalten sauber getrennt. In HTML-Seiten werden Portlets rein deklarativ eingefügt (beim Portal Manager mit Hilfe eines `npspm`-Tags);
- Objektorientierung bringt klare Interfaces und Datenstrukturen mit sich;
- Durch die höhere Modularität kann Code leichter wiederverwendet werden.

2.2 Modulare Konfiguration mit Spring Beans

Die Java-Web-Applikationen von Infopark bestehen aus mehreren Komponenten. Mit Hilfe des Spring-Frameworks können diese Komponenten leicht ausgetauscht und angepasst werden. Infopark liefert zahlreiche Komponenten mit, beispielsweise für die folgenden Aufgaben:

- Ermittlung von Benutzern
- Auslieferung von Inhalten
- Zugriffsschutz
- Authentifizierung

Jede dieser Komponenten ist ein sogenanntes Bean. Über eine oder mehrere XML-Dateien lässt sich die Zusammenstellung der Komponenten für die Web-Applikation festlegen. Dies bedeutet, dass der Funktionsumfang und das Verhalten der Applikation einfach an die Erfordernisse angepasst werden können, indem die Konfigurationsdateien entsprechend geändert werden. Auf diese Weise können auch kundenspezifische Komponenten integriert werden.

Jedes Bean basiert auf einer Java-Klasse. Wird ein Bean verwendet, so erzeugt das Spring Framework ein Objekt der entsprechenden Klasse. Ein Bean wird folgendermaßen eingebunden:

```
<bean id="sessionInvalidator" class="com.infopark.pm.DefaultSessionInvalidator"/>
```

Über die optionale `id` kann das Bean an anderer Stelle referenziert werden.

Falls ein Bean Eigenschaften hat, die an die Installation angepasst werden müssen, können diese auf folgende Weise gesetzt werden:

```
<bean id="httpHelper" class="com.infopark.libs.http.ApacheHttpHelper">
  <property name="connectionTimeout" value="3"/>
  <property name="defaultSocketTimeout" value="60"/>
  <property name="maxConnections" value="10"/>
</bean>
```

Ein Bean kann auf die Funktionalität eines anderen Beans zugreifen. So benötigt beispielsweise das Bean `permissionManager` Zugriff auf Inhalte. Damit man bei der Wahl des Beans, das die Inhalte bereitstellt, flexibel ist, verfügt der `permissionManager` über eine Eigenschaft `documentSource`. Dieser Eigenschaft kann per Referenz das zu verwendende Bean zugewiesen werden. Im folgenden Beispiel wird zunächst das Bean `externalDocumentSource` zur Bereitstellung der Inhalte konfiguriert. Anschließend wird das Bean `permissionManager` eingebunden, das auf die Inhalte über das Bean `externalDocumentSource` zugreift. Die Eigenschaften `permissionReader` und `permissionResolver` werden ebenfalls mit einem Bean belegt. Da das jeweilige Bean nur an dieser Stelle verwendet wird, wird es direkt in der Eigenschaft eingebunden, ohne `id`.

```
<bean id="externalDocumentSource" class="com.infopark.pm.doc.FileDocumentSource">
  <property name="documentRoot"
    value="/opt/infopark/fiona/instance/default/export/online/docs"/>
  <property name="inputEncoding" value="UTF-8"/>
</bean>

<bean id="permissionManager" class="com.infopark.pm.doc.DocumentPermissionManager">
  <property name="documentSource" ref="externalDocumentSource"/>
  <property name="permissionReader">
    <bean class="com.infopark.pm.FionaPermissionReader"/>
  </property>
  <property name="permissionResolver">
    <bean class="com.infopark.pm.FionaPermissionResolver"/>
  </property>
</bean>
```

Die Eigenschaften eines Beans werden von der Klasse bestimmt. Um eine Eigenschaft setzen zu können, muss die Klasse eine öffentliche Methode haben, deren Name sich aus `set` und dem Namen der Eigenschaft zusammensetzt. Der Methode wird genau ein Argument übergeben, der Wert der Eigenschaft. Im obigen Beispiel wird die Methode `setDocumentSource()` der Klasse `com.infopark.pm.doc.DocumentPermissionManager` aufgerufen. Die öffentlichen Methoden der im Lieferumfang von CMS Fiona enthaltenen Java-Klassen können Sie der API-Dokumentation im Verzeichnis `share/doc/javadoc/pm` im Installationsverzeichnis des CMS entnehmen.

Hinweise zur Fehlersuche

Konfigurationsfehler führen dazu, dass die Web-Applikation sich nicht deployen lässt. Die Logdatei der betreffenden Web-Applikation gibt Auskunft über die Art des Fehlers. Bei solchen Fehlern tritt immer eine `BeanDefinitionStoreException` auf. Typische Gründe sind:

- Line ... in XML document ... is invalid

Die Konfigurationsdatei ist kein gültiges XML. Prüfen Sie die Datei auf syntaktische Richtigkeit.

- Bean class [*Klassenname*] not found

  Die für das Bean als `class` angegebene Java-Klasse *`Klassenname`* ist unbekannt. Entweder ist der Klassenname falsch geschrieben, oder die Klasse ist nicht im Verzeichnis `WEB-INF/classes` oder in einem jar-Archiv unter `WEB-INF/lib` in der Web-Applikation zu finden.

- Error setting property values

  Der Wert der Eigenschaft konnte nicht gesetzt werden. Sofern kein Schreibfehler vorliegt, ist die Methode möglicherweise nicht öffentlich (`public`) oder der Wert passt nicht zum Typ des Arguments der Methode.

- Could not instantiate class [*Klassenname*]

  Die angegebene Java-Klasse *`Klassenname`* konnte zwar gefunden werden, es kann jedoch kein Objekt erzeugt werden. Möglicherweise muss eine von davon abgeleitete Klasse verwendet werden.

2.3 Architektur des Portal Managers

2.3.1 Technische Komponenten

Der Infopark Portal Manager ist eine standardkonforme Servlet-Applikation, die in einen Servlet-Container (wie den Infopark Trifork Application Server) deployed wird. Er besteht aus Filtern, Servlets und Beans sowie einigen weiteren unterstützenden Komponenten.

Filter

Filter analysieren und modifizieren eingehende Requests. Sie sind in einer so genannten *Filter Chain* organisiert. Jeder Request durchläuft vor der Beantwortung der Reihe nach einige oder alle Filter. Die Response durchläuft auf dem Weg zurück zum Benutzer noch einmal alle Filter in umgekehrter Reihenfolge. Die Filter können den Request und die Response je nach Aufgabe geeignet ergänzen, verändern oder gar blockieren. Je nach benötigter Funktionalität können Filter ausgetauscht, entfernt oder ergänzt werden. Die wichtigsten im Lieferumfang des Portal Managers enthaltenen Filter sind:

- NTLMFilter
- AuthenticationFilter
- PortletFilter
- ContentFilter
- PermissionFilter

Servlets

Servlets dienen dazu, die Inhalte auszuliefern. Es gibt im Portal Manager ein Servlet für statische Inhalte (Binärdateien) und dynamische Inhalte, das `ContentServlet`.

Filter und Servlets werden – wie bei Java-Webanwendungen üblich – über die Datei `web.xml` konfiguriert.

Beans

Beans sind austauschbare Module, die Filtern und Servlets bestimmte Funktionen zur Verfügung stellen. Sie werden über die Datei `pm.xml` definiert und konfiguriert. Wichtige Beans sind beispielsweise `userManager`, `portletContainer`, `documentManager`, `permissionManager`.

2.3.2 Portal Manager und HTTP-Server

Prinzipiell ist jeder Servlet-Container auch ein Webserver. Dennoch ist es derzeit empfehlenswert, einen HTTP-Server vor den Trifork zu schalten. Wir empfehlen aus Gründen des hohen Bekanntheitsgrades und der Stabilität, den Apache HTTP-Server einzusetzen. Einen HTTP-Server zu verwenden, hat folgende Vorteile:

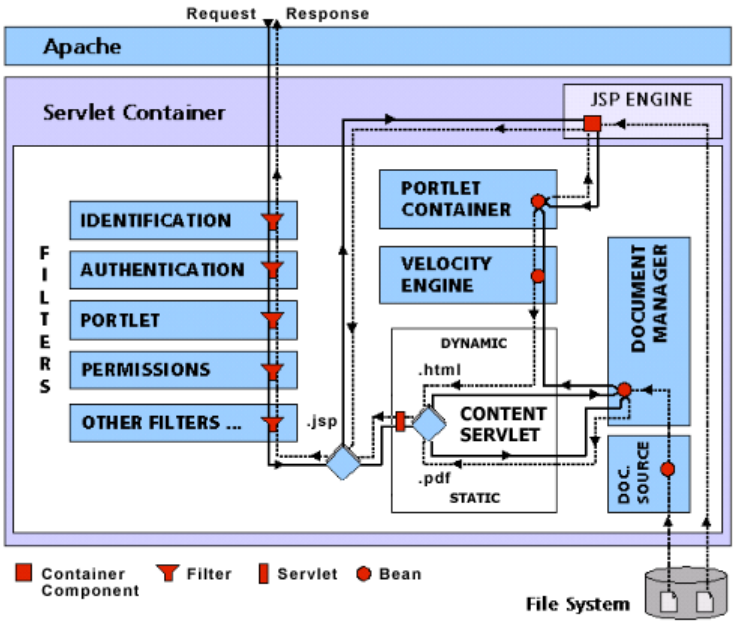
- Es wird möglich, privilegierte Ports wie 80 oder 443 zu nutzen, ohne den Webserver als Superuser starten zu müssen. Dies ist gegenwärtig nur mit einem HTTP-Server möglich.
- Um existierende Logdatei-Analysetools unverändert weiter nutzen zu können, benötigt man Logdateien im commons-logging-Format, die der Apache HTTP-Server erzeugt. Diese Dateien kann der Trifork Application Server derzeit nicht erzeugen. Ab Version 6.5 von Infopark CMS Fiona verfügt der Portal Manager über einen entsprechenden `LoggingFilter`.
- Über den Apache HTTP-Server ist eine schnelle HTTPS-Verschlüsselung zu erreichen.

Gelegentlich gibt es weitere Gründe dafür, einen HTTP-Server vor den Trifork zu schalten:

- Authentifizierung: Über den HTTP-Server können beliebige weitere Authentifizierungsmechanismen einen RemoteUser setzen, der dann vom PM ausgewertet wird.
- Weitere Module wie zur URL-Manipulation, Response-Komprimierung, Bandbreitenbeschränkung o.ä. sind in Verbindung mit dem Portal Manager nutzbar.

2.3.3 Ablauf bei der Beantwortung eines Requests

Im Folgenden wird die Beantwortung eines typischen Requests bei der oben beschriebenen Konfiguration (HTTP-Server, Trifork, Portal Manager) erläutert. Es wird eine HTML-Datei angefordert, in der ein Portlet enthalten ist.



Die obige schematische Darstellung zeigt den Ablauf auch für JSPs und binäre Inhalte.

- Ein Website-Besucher schickt über einen Browser einen HTTP-Request an Port 80 des Webservers.
- Ein Apache nimmt den Request entgegen und leitet ihn über `mod_jk` weiter an Trifork.
- Der Trifork ruft der Reihe nach folgende Filter für den Request auf:
  - Identification beispielsweise via `NTLMFilter`. Dieser ermittelt die Identität des Besuchers und setzt den `RemoteUser`.
  - Der `AuthenticationFilter` erzeugt unter Verwendung des `RemoteUser` mit Hilfe eines Zugriffs auf das ADS ein User-Objekt.
  - Der `PortletFilter` prüft, ob der eingehende Request ein `PortletRequest` ist und leitet ihn gegebenenfalls an den Portlet-Container weiter.
  - Der `PermissionFilter` prüft, ob es Zugriffsbeschränkungen für die angeforderte Datei gibt. Wenn ja, prüft er, ob der Besucher die Datei erhalten darf und bricht die Request-Bearbeitung ab, falls dies nicht der Fall ist.
- Anhand der Konfiguration in der `web.xml` ermittelt der Trifork, dass für den Request das `ContentServlet` zuständig ist.
  - Das `ContentServlet` fordert die Datei vom `DocumentManager` an.
  - Der `DocumentManager` holt sie im Live-Betrieb über die `FileDocumentSource` aus dem Online-Verzeichnisbaum, den die Template-Engine erzeugt hat
  - Das `ContentServlet` ermittelt den MIME-Typ des Dokumentes und liefert den Inhalt statisch oder dynamisch aus, in Abhängigkeit vom MIME-Typ.
  - Eine HTML-Datei wird dynamisch ausgeliefert. Das `ContentServlet` verarbeitet das Dokument als Velocity-Template.
  - Die Velocity-Engine erkennt ein Kommando, mit dem ein Portlet eingefügt wird, und reicht die Anfrage an den `PortletContainer` weiter.
  - Der `PortletContainer` schickt einen Render-Request an das angegebene Portlet und liefert den in der Response enthaltenen HTML-Code zurück.
  - Das `ContentServlet` liefert die zusammengefügte Seite aus.
- Alle Filter haben nun in umgekehrter Reihenfolge noch die Möglichkeit, die ausgehende Antwort zu modifizieren. Der `AuthenticationFilter` nutzt die Gelegenheit und fügt einen Cookie an, der die Authentifizierung beim nächsten Request erleichtert.

2.4 Authentifizierung und Zugriffsschutz

2.4.1 Begriffsbestimmung

Die Authentifizierung bezeichnet den Vorgang der Überprüfung der Identität eines Benutzers. Die Authentisierung hingegen bezeichnet den Vorgang des Nachweises der eigenen Identität.

Bei einer Identitätsüberprüfung oder Identifizierung gibt es daher immer einen Teilnehmer, der sich authentisiert und einen, der diesen authentifiziert. Im Englischen wird zwischen den beiden Begriffen nicht unterschieden, das Wort *authentication* steht für beides.

In einem Computerprogramm werden einer Identität (das heißt einer identifizierten Person) üblicherweise Rechte zugeordnet. Autorisierung bezeichnet den Vorgang, mit dem ein Programm prüft, ob eine bestimmte Identität ein bestimmtes Recht besitzt und damit zum Beispiel eine bestimmte Datei lesen darf.

2.4.2 Möglichkeiten zur Authentifizierung

Die Möglichkeiten des Portal Managers zur Authentifizierung können nahezu beliebig erweitert werden, indem die entsprechenden Filter hinzugefügt werden. Die folgenden Mechanismen werden bereits im Standardlieferumfang unterstützt.

AuthenticationFilter

Der `AuthenticationFilter` in Infopark CMS Fiona unterstützt die folgenden Mechanismen:

- Login und Passwort über Basic-Authentication.
- Login über ein Single-Sign-On-Ticket, beispielsweise für das SSO mit einem SAP Enterprise-Portal.
- Übernahme eines gesetzten `RemoteUser`. Dadurch kann ein beliebiger vorgeschalteter Mechanismus eingesetzt werden.

Single-Sign-On über Windows' NTLM

Der NTLMFilter ermittelt über den Einsatz von NTLM einen `RemoteUser`. Dadurch genügt es beim Einsatz von Windows und des Internet Explorers (eingeschränkt auch Firefox), wenn sich der Benutzer einmal bei Windows anmeldet. Dieses Login steht dann auch im Portal Manager zur Verfügung.

Der NTLMFilter erzwingt eine Authentisierung. Es ist allerdings möglich, einen Fallback auf Basic-Authentication zu aktivieren, falls die NTLM-Authentifizierung fehlschlägt.

Es ist möglich, bestimmte URLs und bestimmte IP-Bereiche von der Authentifizierung auszunehmen.

Behandlung anonymer Benutzer

Kommt kein expliziter Login-Request und ist kein `RemoteUser` gesetzt, wird der Request anonym behandelt. Greift ein anonymer Benutzer auf eine geschützte Datei zu, blockiert der `PermissionFilter` den Request. Der Benutzer erhält anstelle der Datei den HTTP-Response-Code "401 Unauthorized". Dieser wird vom Servlet-Container typischerweise auf eine in der `web.xml` konfigurierte Fehlerseite umgelenkt.

Beim Einsatz des NTLMFilters ist es im Auslieferungszustand des Portal Managers nicht möglich, als anonymer Benutzer auf Inhalte zuzugreifen, da der NTLMFilter eine Authentifizierung erzwingt. Indem jedoch das URL-Muster des Filters (*URL pattern*) in der Filterkonfiguration (diese befindet sich in der `web.xml`) so gesetzt wird, dass der Filter in bestimmten Bereichen der Website nicht greift, können diese Bereiche auch nicht authentifizierten Benutzern zugänglich gemacht werden.

2.4.3 Die Auslieferung geschützter Inhalte verhindern

Die Auslieferung sowohl textueller als auch binärer Inhalte kann auf bestimmte Benutzergruppen beschränkt werden.

Die Prüfung der Zugriffsberechtigung geschieht auf Dateiebene. Typischerweise hat eine Datei entweder keine Zugriffsbeschränkungen (frei verfügbarer Inhalt) oder es ist eine Reihe von Gruppen definiert, die die Datei lesen dürfen (geschützer Inhalt). Im ersten Fall wird die Datei einfach ausgeliefert. Im zweiten Fall wird geprüft, ob es Gründe dafür gibt, dass der Benutzer die Datei lesen darf. Hierfür wird die Liste der Gruppen des Benutzers ermittelt und geprüft, ob mindestens eine davon unter den Gruppen ist, die die Datei lesen dürfen.

Alternativ können beliebige weitere Prüfungen hinzukonfiguriert werden. Mitgeliefert wird bereits eine Prüfung auf der Basis der IP-Adresse des zugreifenden Rechners.

Die Benutzergruppen, die dazu berechtigt sind, eine exportierte Datei zu lesen, werden der entsprechenden Datei im Redaktionssystem (mittels Content Navigator oder per Tcl-Befehl im Content Management Server) zugewiesen. Dieses Recht heißt "Liveserver-Leserecht", das entsprechende Tcl-Schlüsselwort heißt `permissionLiveServerRead`.

2.4.4 Links auf geschützte Inhalte verbergen

Der Portal Manager verfügt über die folgenden beiden Mechanismen zur Unterdrückung von Links auf geschützte Inhalte.

Automatische Linkunterdrückung

In der Template Engine lässt sich konfigurieren, ob Links auf geschützte Inhalte automatisch unterdrückt werden sollen. Ein Link wird dadurch unterdrückt, dass das `href`-Attribut im `a`-Tag entfernt wird. Zur Unterdrückung wird beim Export jedes `href`-Attribut mit Velocity-Code umgeben, der zur Request-Zeit eine Zugriffsprüfung ausführt und das Attribut gegebenenfalls entfernt.

Diese Vorgehensweise hat den Vorteil, dass auch Inhalte, die von Redakteuren erzeugt werden, garantiert keine Links auf geschützte Inhalte enthalten sind.

Sie hat allerdings folgende Nachteile:

- Es wird nur der Link selbst entfernt. Der verlinkte Text (zum Beispiel der Titel des geschützten Dokuments) sowie möglicherweise inhaltlich dazu gehörender umgebender HTML-Code wie eine Tabellenzelle in einem Menü bleiben bestehen.
- Sie ist nur global an- und abschaltbar.
- Sie verlangsamt die Seitenauslieferung, weil für jeden Link auf der ausgelieferten Seite zusätzlicher Velocity-Code ausgeführt werden muss.

Die automatische Linkunterdrückung kann in der Konfiguration in der Datei `export.xml` mit Hilfe des Konfigurationswertes `export.convertNpspm.hideForbiddenLinks` ein- und ausgeschaltet werden. Sie ist voreingestellt aktiviert.

npspm-Elemente

Mit Hilfe von `npspm`-Elementen können Layout-Designer den HTML-Code, mit dem beispielsweise Linklisten angezeigt werden, gezielt anpassen, so dass die relevanten Teile nur sichtbar sind, wenn der Benutzer bestimmte Bedingungen erfüllt:

- `npspm showIfAccessible`
- `npspm showIfLoggedIn`
- `npspm showIfMember`

2.4.5 Geschützte Inhalte aus Suchergebnissen ausschließen

Um zu vermeiden, dass Benutzer über die Suche Dokumente finden können, die sie nicht lesen dürfen, werden die leseberechtigten Gruppen gemeinsam mit den Dokumenten indiziert. Die Suchanfrage kann dann so erweitert werden, dass zusätzlich zum Suchbegriff des Benutzers verlangt wird, dass mindestens eine der Gruppen des Benutzers in den leseberechtigten Gruppen des Dokuments enthalten ist, oder dass das Dokument frei verfügbar ist.

Die folgende Beispiel-Konfiguration für das Such-Portlet (`SearchPortlet`) implementiert eine solche Suchanfrage:

```
#macro (getPermissionQuery)
  #set($groups = "")
  #foreach ( $perm in $user.permissions)
    #if ("$!groups" != "")
      #set ($groups = "$groups , "$perm.trim()"" )
    #else
      #set ($groups = ""$perm.trim()"")
    #end
  #end
  #if ("$!groups" != "")
    <#ANY>((($groups) <#IN> permissionLiveServerRead), (
      "free" <#IN> noPermissionLiveServerRead))
  #else
    <#YESNO>("free" <#IN>noPermissionLiveServerRead)
  #end
#end
...
<condition>
  <and>
    [Hier steht Ihre Search-Query-Formulierung]
    <vql-statement>#getPermissionQuery</vql-statement>
  </and>
</condition>
...
```

Ab Version 6.5 unterstützt der Portal Manager von Infopark CMS Fiona neben der eigentlichen Suchanfrage auch eine Base-Query. Die Base-Query ist der eigentlichen Suchanfrage vorgeschaltet und reduziert die Menge der Dokumente, auf die die Suchanfrage angewendet wird. Sie vereinfacht den Code erheblich.

Die `and`-Sektion der Base-Query wird um Folgendes ergänzt:

```
<or>
  <vql-statement>
    "free" <#IN>noPermissionLiveServerRead
  </vql-statement>
  #foreach ($group in $user.permissions)
    <vql-statement>
      "$group" <#IN>permissionLiveServerRead
    </vql-statement>
  #end
</or>
```

2.5 Portlets bereitstellen

Der Infopark Portal Manager bietet drei Möglichkeiten, Portlets bereitzustellen. Dadurch lassen sich Funktionalitäten und Dienste flexibel integrieren.

2.5.1 Lokale Portlets

Lokale Portlets werden in einer Web-Applikation ausgeführt, die sich im gleichen Application Server wie der Infopark Portal Manager befinden. Im Vergleich zu entfernten Portlets ist diese Art der Bereitstellung am performantesten, da keine Kommunikation über das Netzwerk erforderlich ist.

2.5.2 Portlets in einem entfernten Infopark Portal Manager

Unter folgenden Umständen kann es sinnvoll sein, mehrere Infopark Portal Manager zu betreiben:

- Sicherheitsaspekte: Portlets, die sicherheitskritische Daten verarbeiten, sollten auf einem speziell gegen Angriffe geschützten Server ausgeführt werden.
- Mehrfachnutzung: Bei der Verwendung mehrerer Server an unterschiedlichen Standorten reicht es aus, gemeinsam genutzte Portlets nur an einem Ort zur Verfügung zu stellen. Dies vereinfacht die Administration.
- Lastaspekte: Sehr rechenintensive Portlets können auf einem separaten Server ausgeführt werden, um die Performanz des Gesamtsystems zu verbessern.

2.5.3 Portlets von einem Webservice

Mit dem Infopark Portal Manager können Portlets genutzt werden, die von einem Webservice bereitgestellt werden. Dabei greift der Infopark Portal Manager über HTTP auf einen entfernten Server zu, der die Portlets ausführt.

Der Kommunikation zwischen den Servern liegt die Spezifikation WSRP (Web Services for Remote Portlets) zugrunde. Üblicherweise ist der entfernte Server ein Portal-Server nach JSR168. Die Seite, die Portlets einbindet, wird Consumer genannt, diejenige, die Portlets gemäß WSRP-Spezifikation bereitstellt, Producer.

WSRP schafft also eine gemeinsame Basis für die Portal-Funktionalität unterschiedlicher Herkunft. Durch die Möglichkeit, den Infopark Portal Manager als WSRP-Consumer einzusetzen, können auch Portlets, die auf anderen Sytemen wie beispielsweise im IBM Websphere Portal Server laufen, genutzt werden.

2.6 Voraussetzungen

Der Portal Manager von Infopark CMS Fiona kann ab Version 6.5.1 als Consumer nach WSRP-Standard 1.0 eingesetzt werden.

Folgende Vorausetzungen müssen erfüllt sein, um Remote Portlets im Infopark Portal Manager einsetzen zu können:

- Es muss ein Portalserver existieren, der Remote Portlets nach WSRP-1.0 Spezifikation bereitstellen kann und als Producer dient.
- Portlets auf diesem Portalserver, die über Webservices angeboten werden, müssen vorhanden sein.
- Der Infopark Portal Manager und der Producer müssen über HTTP/HTTPS miteinander kommunizieren können.
- Die den Service beschreibende WSDL-Datei sowie alle darin inkludierten Schemata und Namensräume müssen für den Infopark Portal Manager erreichbar sein, auch wenn sie auf entfernten Rechnern liegen.
- Die URL der WSDL-Datei für den Service muss bekannt sein.
  Beispiel:
  `http://portalstandards.oracle.com/portletapp/portlets?WSDL`

2.7 Einschränkungen

Obwohl eine WSRP-Spezifikation existiert, könnten einige Hersteller proprietäre Funktionen in ihre Produkte aufgenommen haben. Gegenwärtig unterstützt der Infopark Portal Manager keine proprietären Funktionen.

Der Infopark Portal Manager bietet eine Umsetzung der WSRP-Spezifikation, die die folgenden Einschränkungen aufweist:

- Keine Unterstützung von *Producer Mediated Sharing* (`CookieProtocol:perGroup`)
- Keine Unterstützung von Ankern (`wsrp-fragmentID`)

3

3 Den Portal Manager verwenden

3.1 Zugriffssteuerung und Personalisierung

Die Zugriffsteuerung ist eine von mehreren Möglichkeiten, um dem Besucher einer Website auf ihn zugeschnittene Inhalte anzubieten oder die Auslieferung von Inhalten zu unterbinden. Das Konzept wird häufig ausschließlich als Moglichkeit der *Einschränkung der Zugriffsmöglichkeiten von Besuchern* verstanden, obwohl die selektive Ausgabe unterschiedlicher Inhalte ein mindestens ebenso wichtiges Ziel der Zugriffssteuerung ist.

Zugriffssteuerung setzt die Möglichkeit voraus, Leserechte für Live-Dokumente vergeben zu können. Diese Möglichkeit existiert im Redaktionssystem, wo dem zu jeder Datei gehörenden Live-Server-Leserecht-Feld eine Reihe von Benutzergruppen zugeordnet werden kann.

Der Portal Manager liefert Inhalte unter Berücksichtigung der erteilten Live-Server-Leserechte aus. Wird der Zugriff für ein Dokument im CMS auf bestimmte Live-Benutzergruppen beschränkt, so erlaubt der Portal Manager den Zugriff auf das Dokument erst nach erfolgreicher Authentifizierung. Links auf dieses Dokument werden gegebenfalls deaktiviert.

Dateien können jedoch nicht nur komplett mit Leserechten versehen werden, sondern auch partiell. Hierfür steht ein spezielles Sprachelement, `<npspm>`, zur Verfügung, das in den Inhalten selbst eingesetzt werden kann. Das `npspm`-Element wird wie ein gewöhnliches HTML-Element in Layouts eingefügt.

Ähnlich wie `npsobj`-Elemente während des Exports in HTML-Elemente übersetzt werden, werden auch `npspm`-Elemente übersetzt, und zwar in eine serverseitig ausgewertete Sprache. Eine der unterstützten Sprachen ist Velocity, da der Portal Manager eine Velocity Engine enthält. Der Portal Manager selbst ist eine Web-Applikation, die in einem Servlet-Container läuft, der eine JSP Engine enthält. Daher ist JSP eine weitere mögliche Sprache, in die `npspm`-Elemente während des Exports übersetzt werden können (siehe Architektur des Portal Managers). Eine PHP-Seite kann keine Funktionalitäten des Portal Managers wie Zugriffsschutz oder Portles enthalten, da serverseitig immer nur eine Sprache pro Datei ausgewertet werden kann.

Die Sprache, in die die `npspm`-Elemente während des Exports übersetzt werden, ist abhängig von der Dateierweiterung. Die Zuordnung einer Sprache zu Dateierweiterungen kann mit dem Parameter `export.convertNpspm.mapping` in der Datei `export.xml` konfiguriert werden. Aus dem oben genannten Grund können dort neben `velocity` und `jsp` keine anderen Werte für serverseitige Sprachen angegeben werden.

Verwenden Sie bitte für die Zugriffssteuerung und Personalisierung die `npspm`-Elemente und nicht die Übersetzung nach `Velocity` oder `JSP`. Dadurch stellen Sie sicher, dass auch in neuen, weiterentwickelten Fiona-Versionen immer die passende Übersetzung verwendet wird.

Die unterschiedlichen Funktionen für die Zugriffssteuerung sind über die Tag-Attribute des `npspm`-Elements zugänglich.

3.1.1 includePortlet

Aufgabe

Mit diesem Element wird ein Portlet in einem HTML-Dokument referenziert. Mit Hilfe des Instanzenbezeichners können bei Bedarf unterschiedliche Instanzen des Portlets generiert werden. Möchten Sie beispielsweise ein Portlet einbinden, mit dem Ihre Website bewertet werden kann, so ist eine Instanz ausreichend und der Instanzenname braucht nicht angegeben zu werden, selbst wenn das Portlet auf mehr als einer Seite eingebunden wird. Verwenden Sie dieses Portlet jedoch, um einzelne Seiten bewerten zu lassen, so muss für jede Seite eine eigene Portlet-Instanz verwendet werden.
In diesem Fall kann beispielsweise der Pfad der Seite als Instanzenname verwendet werden, da er eindeutig ist. Der Pfad kann mit Hilfe einer @-Referenz ermittelt werden (siehe das Beispiel unten).

Syntax

```
<npspm includePortlet="urlPath" instance="instanceId" language="lang"
 withBorder="borderFlag" />
```

Wenn das einzufügende Portlet in der gleichen Web-Applikation liegt wie der Portal Manager, ist `urlPath` der Name des Portlets oder der Alias-Pfad, der als `portletPathMapping` in der Datei `pm.xml` angegeben wurde. Dieser Name darf nicht / sein. Liegt das Portlet dagegen in einer anderen Web-Applikation, ist `urlPath` der URL-Pfad des Portlets, bezogen auf das Web-Applikationen-Verzeichnis (normalerweise `webapps`).

Mit dem Attribut `instanceId` kann ein Instanzenbezeichner angegeben werden, um mehrere gleiche Portlets auf einer Website verwenden zu können. Der Bezeichner kann aus beliebigen Zeichen bestehen.

Die Sprache, in der ein Portlet sich anzeigt, entspricht normalerweise der Sprache, die der Benutzer im Browser eingestellt hat. Mit dem Parameter `language` kann diese Voreinstellung übergangen werden.

Der Wert von `language`, *`lang`*, ist ein Kürzel wie `de` oder `en`, das für eine der vom Portlet unterstützten Sprachen steht. Diese Sprachen sind in der Datei `portlet.xml` definiert, die sich im `WEB-INF`-Verzeichnis der Portlet-Web-Applikation befindet.

`withBorder` legt mit den Werten `true` und `false` für *`borderFlag`* fest, ob das Portlet mit einem Rahmen, Titelzeile und Buttons in der Titelzeile ausgegeben werden soll.

Beispiel

```
<npspm includePortlet="/myportlets/ranking" instance="@visiblePath" withBorder="false" />
```

3.1.2 includePage

Aufgabe

Das Element wird durch den analysierten ("geparsten") Inhalt der referenzierten Seite ersetzt, sofern der aktuelle Benutzer auf diese Seite lesend zugreifen darf. Andernfalls wird das Element

ignoriert. Um eine endlose Rekursion bei zyklischen Einfügungen zu vermeiden, ist die maximale Verschachtelungstiefe auf 10 begrenzt.

Syntax

```
<npspm includePage="path" />
```

Als *path* wird ein interner Pfad angegeben, der auf eine Datei vom Typ `document` (Dokument) oder `publication` (Ordner) verweist. Das Element hat keinen Inhalt.

Enthält die Seite mit dem angegebenen Pfad interne Links, so verweisen diese – nachdem die Seite inkludiert wurde – möglicherweise nicht mehr auf das gewünschte Ziel. Der Grund hierfür ist, dass Pfade in Links relativ sind. Wird beispielsweise die Seite C in A und B inkludiert, wobei A und B nicht im gleichen Ordner liegen, müsste C relativ zu zwei Positionen in der Ordnerhierarchie sein, um auf das gleiche Ziel zu zeigen.

Dieser Effekt kann dadurch umgangen werden, dass in inkludierten Dateien absolute Links verwendet werden. Allerdings werden für die Vorschau und den Live-Auftritt Links mit unterschiedlichen Präfixen benötigt.

Dies lässt sich mit Hilfe von Velocity-Code (der zur Laufzeit ausgewertet wird) lösen. Das Tool `$document` hat die Methode `getUrl`, die einen absoluten Pfad um den jeweiligen Präfix ergänzt und das Ergebnis in eine URL umwandelt. Das folgende Beispiel zeigt dies anhand einer Liste von Links, die mit einer `toclist` erzeugt wird. Der Code, der die URL erzeugt, wird zunächst in einer Export-Variablen abgelegt, deren Wert dann im `href`-Attribut per @-Referenz ausgelesen wird:

```
<npsobj list="toclist">
  <npsobj modifyvar="set" name="robustPath">$document.getUrl(
    <npsobj insertvalue="var" name="visiblePath"/>
  )</npsobj>
  <a href="@robustPath"><npsobj insertvalue="var" name="title"/></a>
</npsobj>
```

Durch diese Vorgehensweise wird der absolute Pfad des Linkziels (bezogen auf die Ordnerhierarchie) zum Zeitpunkt des Exports ermittelt und mit Code versehen, der zur Laufzeit diesen Pfad in die richtige URL umwandelt.

Beispiel

```
<npspm includePage="/intranet/docs/public/apps" />
```

3.1.3 showIfAccessible

Aufgabe

Der Inhalt dieses Elements wird genau dann angezeigt, wenn der am Portal angemeldete Benutzer (oder der Default-User, falls niemand angemeldet ist) das Liveserver-Leserecht (`permissionLiveServerRead`) für die Seite hat, die durch `path` referenziert wird. Mit `negate="true"` kann diese Bedingung umgekehrt werden, d. h. die Seite wird nur dann angezeigt, wenn der Benutzer das Liveserver-Leserecht nicht hat.

Syntax

```
<npspm showIfAccessible="path" [negate="negateFlag"]>content</npspm>
```

Als *path* wird ein Pfad zu einer CMS-Datei (ein interner Pfad) angegeben. Für *negateFlag* können die Werte `true`, `yes`, `on`, `1` (für wahr) und `false`, `no`, `off`, `0` (für falsch) eingesetzt werden. Das Attribut `negate` ist optional.

Beispiel

```
<npspm showIfAccessible="/intranet/docs">
  <a href="/intranet/docs">Unsere Dokumente</a>
</npspm>
<npspm showIfAccessible="/intranet/docs" negate="true">
  Sie haben keinen Zugriff auf unsere Dokumente
</npspm>
```

3.1.4 showIfMember

Aufgabe

Der Inhalt dieses Elements wird genau dann angezeigt, wenn der am Portal angemeldete Benutzer (oder der Default-User, falls niemand angemeldet ist) Mitglied mindestens einer der angegebenen Gruppen ist. Der Inhalt des Elements wird auch dann angezeigt, wenn keine Gruppe angegeben ist. Mit `negate="true"` kann diese Bedingung umgekehrt werden, d. h. die Seite wird nur dann angezeigt, wenn Gruppen angegeben wurden und der Benutzer in keiner dieser Gruppen Mitglied ist.

Syntax

```
<npspm showIfMember="group1|...|groupN" [negate="negateFlag"]> ... </npspm>
```

Als *group1* bis *groupN* werden durch vertikale Striche getrennte Gruppennamen angegeben. Es ist auch möglich, keinen Gruppennamen anzugeben. Für *negateFlag* können die Werte `true`, `yes`, `on`, `1` (für wahr) und `false`, `no`, `off`, `0` (für falsch) eingesetzt werden. Das Attribut `negate` ist optional.

Beispiel

```
<npspm showIfMember="users|admins">
  Sie sind Mitglied der Gruppe "users" oder "admins" oder beider Gruppen!
  <npspm showIfMember="users" negate="true">
    Sie gehören nicht zur Gruppe "users", also sind Sie ein Admin!
  </npspm>
</npspm>
```

3.1.5 showIfLoggedIn

Aufgabe

Dieses Element bewirkt, dass sein Inhalt nur dann angezeigt wird, wenn ein angemeldeter Benutzer (`showIfLoggedIn="true"`) bzw. nicht angemeldeter Benutzer (`showIfLoggedIn="false"`) die Seite angefordert hat.

Syntax

```
<npspm showIfLoggedIn="knownFlag"> ... </npspm>
```

Für *knownFlag* können die Werte `true`, `yes`, `on`, `1` (für wahr) und `false`, `no`, `off`, `0` (für falsch) eingesetzt werden.

```
<npspm showIfLoggedIn="true">
  Sie sind am Portal angemeldet!
</npspm>
```

3.1.6 showIfLanguage

Aufgabe

Mit der `npspm`-Anweisung `showIfLanguage` können abhängig von der Sprache des Benutzers Teile des Inhalts ein- oder ausgeblendet werden.

Syntax

```
<npspm showIfLanguage="lang"> ... </npspm>
```

Für *lang* kann eines der aus zwei Zeichen bestehenden Sprachkürzel wie `de`, `en`, `fr` usw. angegeben werden.

Die Bestimmung der Sprache des Benutzers ist abhängig von der Konfiguration. Ist diese für einen Host nicht eindeutig festgelegt, so kann sie durch die Browsereinstellungen des Benutzers, ein Attribut des eingeloggten Benutzers und den Requestparameter `lang` beeinflusst werden. Das Verhalten wird über den `LanguageFilter` in der Datei `WEB-INF/web.xml` konfiguriert.

Beispiel

```
<npspm showIfLanguage="de">Deutscher Inhalt</npspm>
<npspm showIfLanguage="de" negate="true">English content</npspm>
```

Für deutschsprachige Benutzer erscheint "Deutscher Inhalt" während alle anderen Benutzer "English content" sehen.

3.1.7 npspm-Elemente parametrisieren

In Layoutdateien des Content Management Servers können alle `npspm`-Anweisungen bis auf `showIfLoggedIn` parametrisiert werden. Dies ist in Layouts erforderlich, in denen `npspm`-Elemente für die einzelnen Bestandteile einer Liste erzeugt werden, die erst beim Export vorliegt (beispielsweise eine `toclist` oder eine Liste freier Links). Eine `npspm`-Anweisung zu parametrisieren bedeutet, Variablen anstelle der Namen bekannter Felder der exportierten Datei zu verwenden. Dadurch lässt sich auch in erzeugten Listen für jedes Element die Zugriffsberechtigung prüfen:

```
<npsobj list="toclist">
  <npspm showIfAccessible="@self"> ... </npspm>
</npsobj>
```

Um nur dann HTML-Text (oder andere Ausgaben) zu erzeugen, wenn ein Benutzer Mitglied einer oder mehrerer Benutzergruppen ist, kann die folgende `npspm`-Anweisung verwendet werden:

```
<npspm showIfMember="@memberList"> ... </npspm>
```

`memberList` stellt im obigen Beispiel ein Mehrfachaufzählungsfeld dar, das Gruppennamen enthält. Wie bei showIfMember beschrieben, können die Gruppennamen auch direkt angegeben werden (mit dem vertikalen Strich als Trenner). Hier das ausformulierte Beispiel, mit dem eine Liste erzeugt wird, bei der für jedes Element die Zugriffsberechtigung geprüft wird:

```
<ul>
  <npsobj list="toclist">
    <npspm showIfMember="@memberList">
      <li>
        <npsobj insertvalue="anchor" name="self">
          <npsobj insertvalue="var" name="title" />
        </npsobj>
      </li>
    </npspm>
  </npsobj>
</ul>
```

In der erzeugten Liste sind nur Elemente enthalten, auf die der Benutzer zugreifen darf.

3.1.8 Fehlerbehandlung bei falschen npspm-Tags

Kann eine `npspm`-Anweisung in einem Dokument nicht interpretiert werden, weil sie fehlerhaft ist (d.h. beispielsweise einen nicht definierten Attributwert enthält), so wird statt des Dokuments eine Fehlermeldung angezeigt.

Zusätzlich wird zum Zwecke der leichten Auffindbarkeit des Fehlers der Fehlerort in die Logdatei der jeweiligen Applikation geschrieben (siehe das `log`-Verzeichnis der jeweiligen Applikation).

3.2 News und News-Feeds im Redaktionssystem erzeugen

3.2.1 Funktionen zur Erzeugung von News-Feeds

Zur Erzeugung von News-Feeds dienen im CMS die folgenden Funktionen:

- In den Systemeinstellungen können Channels definiert werden. Channels dienen dazu, Nachrichten thematisch zu gruppieren, also Dateien in Bezug auf deren Inhalte zu kategorisieren – ähnlich wie ein benutzerdefiniertes Feld `Warengruppe` eine Datei bezüglich der in den Inhalten beschriebenen Produkte kategorisieren würde. Die Channel-Einstellungen sind mit dem Content Navigator über das Menü *Extras* > *Systemeinstellungen* zu erreichen.
- Die Versionen von Dateien des Typs *Ordner* und *Dokument* haben das fest eingebaute Feld `channels`. Das Feld `channels` ist vom Typ *Mehrfachauswahl*, und man kann es daher mit beliebig vielen der konfigurierten Channels belegen. Auf diese Weise lassen sich Inhalte und eine Auswahl von Channels einander zuordnen.
- Dateivorlagen haben das Feld `canCreateNewsItem` (als News auf dem Live-Server bereitstellen). Ist dieses Feld bei einer Vorlage aktiviert, so werden Dateien mit dieser Vorlage bei der Freigabe in eine interne Newsliste aufgenommen, sofern das `channels`-Feld der freigegebenen Version nicht leer ist (dieses Feld also mindestens einen Channel enthält).

- Es können Listen von News erzeugt werden, die sich in beliebig festlegbaren Channels befinden. Hierfür stehen eine NPSOBJ-Anweisung und ein Tcl-Befehl zur Verfügung. Die NPSOBJ-Anweisung liefert analog zur `toclist`-Anweisung eine Kontextliste, so dass die Felder der Version ausgelesen werden können. Der Tcl-Befehl lautet `news`.

3.2.2 Verwendung der NPSOBJ-newslist-Anweisung

Zur Erzeugung von Newslisten in Layouts kann die NPSOBJ-`newslist`-Anweisung auf die folgenden drei Arten verwendet werden:

- Die Liste aller News in allen Channels erzeugen:

```
<npsobj newslist="all" length="20">
  Für jede News evaluierter Text
</npsobj>
```

- Die Liste der News erzeugen, die mindestens einem Channel zugeordnet sind, die als Wert in einem Feld des aktuellen Dokuments enthalten sind:

```
<npsobj newslist="selected" name="Channel-Feld" length="20">
  Für jede News der indirekt angegebenen Channels evaluierter Text
</npsobj>
```

  Das Feld muss einen der Typen *Zeichenkette*, *Text*, *Auswahl* oder *Mehrfachauswahl* haben. Bei den Typen *Zeichenkette* und *Text* müssen die Werte kommasepariert angegeben sein. Bei Feldern vom Typ `Auswahl` oder *Mehrfachauswahl* werden die Werte des Feldes so wie sie sind als Channels verwendet.
- Die Liste der News erzeugen, die direkt angegebenen Channels zugeordnet sind:

```
<npsobj newslist="selected" value="ch1, ch2, ..." length="20">
  Für jede News der direkt angegebenen Channels evaluierter Text
</npsobj>
```

Auch wenn eine News mehr als einem der direkt oder indirekt angegebenen Channels zugeordnet ist, ist sie höchstens einmal in der generierten Newsliste enthalten.

Erzeugte Newslisten enthalten nur News, deren Erscheinungsdatum zum Zeitpunkt des Exports in der Vergangenheit liegt. Intern enthält die Newsliste natürlich alle Dateien, denen ein Channel zugewiesen wurde und deren Vorlage über das Feld `canCreateNewsItem` die Bereitstellung der Datei als News vorsieht. Das Erscheinungsdatum entspricht initial dem Anlegezeitpunkt der Newsdatei, der bei Bedarf über das Gültig-ab-Feld der Arbeitsversion beeinflusst werden kann.

Ein versehentlich freigegebener Newsartikel kann daher aus der generierten Newsliste entfernt werden, indem ihr Erscheinungsdatum in die Zukunft verlegt wird. In der internen Newsliste bleibt sie jedoch erhalten, sofern die News-Datei nicht gelöscht oder ihr Channels-Feld nicht geleert wird. Eine Datei wird auch aus der internen Newsliste entfernt, wenn die Channels, denen die News zugeordnet ist, aus der Systemkonfiguration gelöscht werden. Der Wert des `channels`-Feldes der Versionen wird dadurch jedoch nicht beeinflusst, kann danach also Namen nicht existenter Channels enthalten.

Newsartikel, die nicht existierenden Channels zugewiesen sind, werden nicht wieder in die Newsliste eingetragen, wenn ein fehlender Channel wieder angelegt wird.

Wenn die Template Engine eingesetzt wird, werden alle Dateien, die eine NPSOBJ-`newslist`-Anweisung enthalten, bei jeder Änderung an Dateien und an der Channel-Konfiguration neu exportiert, damit die erzeugten Newslisten stets aktuell sind. (Die Dateien erhalten eine `usesAll`-Abhängigkeit.

3.2.3 Beispiele

Die Liste der 10 neuesten News erzeugen

- Den Channel `sitenews` anlegen.
- Die Dateivorlage `newsitem` anlegen und darin `canCreateNewsItems` (*Auf dem Live-Server als News bereitstellen*) aktivieren.
- An beliebiger Stelle in der Ordnerhierarchie eine Datei `news1` mit der Vorlage `newsitem` anlegen.
- In das Layout der Startseite etwa Folgendes einfügen:

```
<ul>
  <npsobj newslist="all" lenght="10">
    <li>
      <npsobj insertvalue="anchor" destination="self">
        <npsobj insertvalue="var" name="title" />
      </npsobj>
  </npsobj>
</ul>
```

Einen RSS-Feed für Politik-News einrichten

- Den Channel `politics` anlegen;
- Die Dateivorlage `newsitem` anlegen und darin `canCreateNewsItems` (*Auf dem Live-Server als News bereitstellen*) aktivieren.
- Das Feld `description` und die Dateivorlage `feed` anlegen und zu dieser `description` hinzufügen. `canCreateNewsItems` nicht aktivieren.
- Den Ordner `rssfeeds` anlegen.
- Darin ein Basislayout anlegen, das einen RSS-Feed ausgibt (s.u.).
- In dem gleichen Ordner die Datei `politicsfeed` mit der Vorlage `feed` anlegen und in die `description` "Neues aus der Firmenpolitik" eingeben. Als Channels `politics` auswählen.
- Dem Basislayout den folgenden Haupttext geben:

```
<?xml version="1.0" encoding="utf-8"?>
  <!DOCTYPE rss PUBLIC "-//Netscape Communications//DTD RSS 0.92//EN"
"http://my.netscape.com/publish/formats/rss-0.92.dtd">
<rss version="0.92">
<channel>
  <description>
    <npsobj insertvalue="var" name="description" />
  </description>
```

```
  <language>de</language>
  <title><npsobj insertvalue="var" name="title" /></title>
  <link>http://www.mysite.com/</link>
  <copyright>Copyright by Infopark AG</copyright>
  <generator>NPS 6.0</generator>
  <ttl>60</ttl>
  <npsobj newslist="selected" name="channels" length="20">
    <item>
      <title><npsobj insertvalue="var" name="title"/></title>
      <link>http://www.infopark.de<npsobj
        insertvalue="var" name="visiblePath" />
      </link>
      <description>
        <npsobj insertvalue="var" name="description"/>
      </description>
    </item>
  </npsobj>
</channel>
</rss>
```

Für eine RSS-Datei in einer anderen Version (beispielsweise 2.0) kann die Layoutdatei entsprechend angepasst werden.

Um einen weiteren Feed mit Sportnachrichten mit der obigen Layoutdatei zu generieren:

- Den Channel `sports` einrichten.
- Im Ordner `rssfeeds` die Datei `sportsfeed` mit der Vorlage `feed` anlegen und in dessen Feld `description` "Neues aus dem Sport" eingeben. Als Channels `sports` auswählen.
- Jetzt Dateien mit dem Format `newsitem` anlegen und als Channel `sports` auswählen. Sie erscheinen im Sports-Newsfeed.

Einen Newsletter versenden

- Den Ordner `newsletters` anlegen.
- Ein Basislayout schreiben, die den Inhalt des Newsletters erzeugt, indem sie beispielsweise Felder aus entsprechenden Dateien ausliest.
- Das Dateiformat `newsletter` anlegen.
- Im Ordner `newsletters` die Datei `politics` mit dem Format `newsletters` anlegen.
- In den Hauptinhalt von `politics` die Email-Adressen mit einer Adresse pro Zeile eintragen.
- Einen schreiben, der das Feld `exportBlob` von `politics` an jede Email-Adresse im Haupttext von `politics` verschickt.

4

4 Den Portal Manager installieren und konfigurieren

4.1 Den Portal Manager mit einem anderen Application Server betreiben

Bei der Installation von Infopark CMS Fiona wird der Trifork Application Server ebenfalls installiert. In diesem Applikation Server laufen unter anderem das ebenfalls mitgelieferte GUI und der Infopark Portal Manager. Für den Betrieb dieser Web-Applikationen ist daher keine weitere Software von Drittanbietern erforderlich.

Der Portal Manager und die von Infopark mitgelieferten Portlets – nicht jedoch das GUI – können auch in einem anderen Application Server betrieben werden, sofern die folgenden Voraussetzungen erfüllt sind.

1. Der Application Server muss J2EE 1.4 vollständig unterstützen.
2. Der Application Server muss das optionale Feature *cross-context dispatching* unterstützen und es muss aktiviert sein.

4.1.1 Den Portal Manager mit Tomcat 5.x betreiben

Gehen Sie bitte nach der folgenden Anleitung vor, wenn Sie den Portal Manager mit dem Application Server Tomcat, Version 5.5, betreiben möchten. Bei Tomcat der Version 5.0 ist die Vorgehensweise ähnlich, Sie benötigen jedoch das Installationspaket für Tomcat 5.0. Es ist nicht erforderlich, das Kompatibilitätspaket für die Verwendung des JDK 1.4 zu installieren.

1. Laden Sie Tomcat herunter.
    1. Laden Sie das Paket `apache-tomcat-5.5.17.tar.gz` von der Apache Tomcat Website herunter.
    2. Wenn Tomcat mit einem JDK 1.4 betrieben werden soll, so laden Sie auch das Paket `apache-tomcat-5.5.17-compat.tar.gz` herunter.
2. Installieren Sie Tomcat
    1. Entpacken Sie das Paket aus 1.1. Das entstehende Verzeichnis wird im Weiteren als *`tomcatInstallDir`* referenziert.
    2. Wenn Tomcat mit einem JDK 1.4 betrieben werden soll, so entpacken Sie auch das Paket aus 1.2. Verwenden Sie hierbei das gleiche Zielverzeichnis wie für das Tomcat-Package, da zusätzliche jar-Dateien in das Verzeichnis *`tomcatInstallDir`*`/common/lib` gelegt werden.
3. Passen Sie die Umgebungsvariable `PATH` so an, dass das zu verwendende JDK gefunden wird.
4. Sofern auf der gleichen Maschine ein Trifork-Server installiert ist, der auf den Standardports läuft, passen Sie in der Datei *`tomcatInstallDir`*`/conf/server.xml` drei Ports an:

```
<Connector port="8080" maxHttpHeaderSize="8192"
  maxThreads="150" minSpareThreads="25"
  maxSpareThreads="75"
  enableLookups="false" redirectPort="8443"
  acceptCount="100"
  connectionTimeout="20000" disableUploadTimeout="true" />

<Connector port="8009"
  enableLookups="false" redirectPort="8443"
  protocol="AJP/1.3" />

<Connector port="8082"
  maxThreads="150" minSpareThreads="25"
  maxSpareThreads="75"
  enableLookups="false" acceptCount="100"
  connectionTimeout="20000"
  proxyPort="80" disableUploadTimeout="true" />
```

5. Kopieren Sie das *JavaBeans™ Activation Framework* von *fionaInstallDir*/share/lib/activation.jar nach *tomcatInstallDir*/common/lib.
6. Kopieren Sie *JavaMail* von *fionaInstallDir*/share/lib/mail.jar nach *tomcatInstallDir*/common/lib.
7. Kopieren Sie die Web-Applikationen PM und PM-PL aus dem Verzeichnis *fionaInstallDir*/instance/*myInstance*/webapps rekursiv nach *tomcatInstallDir*/webapps/.
8. Passen Sie in den Dateien *tomcatInstallDir*/webapps/PM/WEB-INF/web.xml und *tomcatInstallDir*/webapps/PM-PL/WEB-INF/web.xml den Pfad zur Lizenzdatei license.xml im config-Verzeichnis der CMS-Instanz an. Sofern das CMS auf einem anderen Server läuft, kopieren Sie die Lizenzdatei an einen für den Tomcat erreichbaren Ort.
9. Tragen Sie in den Dateien *tomcatInstallDir*/webapps/PM/WEB-INF/logging.xml und *tomcatInstallDir*/webapps/PM-PL/WEB-INF/logging.xml den Pfad zur Protokolldatei der Web-Applikationen ein, etwa:

   ```
   <param name="File" value="tomcatInstallDir/logs/PM.log"/>
   ```

10. Löschen Sie in den Verzeichnissen META-INF der beiden Web-Applikation PM und PM-PL die Datei trifork-app-conf.xml. Legen Sie in beide Verzeichnisse die Datei context.xml mit dem folgenden, an die Web-Applikation angepassten Inhalt. Für PM:

    ```
    <Context docBase="${catalina.home}/webapps/PM" path="/PM" crossContext="true" />
    ```

    Für PM-PL:

    ```
    <Context docBase="${catalina.home}/webapps/PM-PL" path="/PM-PL" crossContext="true" />
    ```

11. Bereiten Sie PM und PM-PL für den Livebetrieb vor:

    1. Wenn Sie den Portal Manager der Version 6.0.x einsetzen, deaktivieren Sie bitte in der Datei *tomcatInstallDir*/webapps/PM/WEB-INF/pm.xml im Bean documentManager das Bean com.infopark.pm.PreviewDocumentSource (auskommentieren)
    2. Aktivieren Sie das Bean com.infopark.pm.FileDocumentSource (Kommentarzeichen entfernen). Tragen Sie im Property documentRoot den Pfad zum Exportverzeichnis ein.

3. Aktivieren Sie in der Datei *tomcatInstallDir*/webapps/PM/WEB-INF/pm.xml im Bean userManager das Bean mit der zu verwendenden Benutzerverwaltung und konfigurieren Sie sie. Deaktivieren Sie andere Benutzerverwaltungen.
4. Deaktivieren Sie in der Datei *tomcatInstallDir*/webapps/PM-PL/WEB-INF/pm.xml das Bean searchEngine mit der Klasse com.infopark.libs.search.cm.AdvancedCmSearchEngine und aktivieren Sie das gleichnamige Bean der Klasse com.infopark.libs.search.ses.SesSearchEngine. Passen Sie die Propertys host und port entsprechend der Fiona-Installation an.
5. Ändern sie in der Datei *tomcatInstallDir*/webapps/PM-PL/WEB-INF/pm.xml das portletPathMapping zu /PM-PL = /PM-PL.

12. Starten Sie den Tomcat-Server, indem Sie *tomcatInstallDir*/bin/startup.sh ausführen. Sie können nun Ihre exportierten Seiten mit Portlets unter der folgenden URL erreichen:

```
http://myTomcatHost:myTomcatPort/PM/
```

13. Bei Problemen prüfen Sie bitte die Protokolldateien im Verzeichnis *tomcatInstallDir*/logs. Bei allgemeinen Fehlern sind die Dateien catalina.out und catalina.*yyyy-mm-dd*.log aufschlussreich. Bei speziellen Fehlern der einzelnen Web-Applikationen untersuchen Sie bitte die in Schritt 9 konfigurierten Dateien.

4.1.2 Den Portal Manager mit WebSphere Application Server 5.1 betreiben

Bitte gehen Sie folgendermaßen vor, um den Infopark Portal Manager im WebSphere Application Server 5.1 zu betreiben.

1. Kopieren Sie die Lizenzdatei license.xml nach webapps/PM/WEB-INF.
2. Setzen Sie in der Konfigurationsdatei webapps/PM/WEB-INF/web.xml den Wert des Kontextparameters licenseFile auf /WEB-INF/license.xml.
3. Speichern Sie den Inhalt des Verzeichnisses webapps/PM in einer ZIP-Datei, die Sie PM.war nennen.
4. Deployen Sie das Archiv PM.war über die *WebSphere Administrative Console* unter /PM.

Bitte führen Sie die obigen Schritte analog auch für webapps/PM-PL durch.

4.2 Standardkonfiguration

4.2.1 Definition der Servlets und Filter

Der Portal Manager kann in den beiden Dateien web.xml und pm.xml konfiguriert werden. Diese Dateien sind im Verzeichnis WEB-INF unterhalb des Web-Applikationsverzeichnisses des Portal Managers zu finden. In der CMS-Standardinstallation lautet der Pfad (relativ zum CMS-Verzeichnis):

```
instance/default/webapps/PM/WEB-INF
```

In der Datei web.xml werden (wie in jeder anderen Web-Applikation auch) die verwendeten Servlets, Servlet-Filter etc. deklariert und deren URL-Schemata festgelegt.

4.2.2 Portal-Manager-Konfiguration

Die eigentliche Konfiguration des Portal Managers ist in der Datei `pm.xml` zu finden. In dieser Datei sind Spring-Beans konfiguriert. In der Standardkonfiguration sind die folgenden obligatorischen Beans enthalten:

- `hostConfig` (`com.infopark.pm.HostConfig`)
  Legt fest, für welche Hosts Anfragen angenommen werden und welche Sprachen unterstützt werden, siehe auch `<npspm showIfLanguage>` und die Portletkonfiguration in der Datei `WEB-INF/portlet.xml`.
- `userManager` (`com.infopark.pm.user.UserManager`)
  Die Benutzerverwaltung des Systems, über die der `AuthenticationFilter` auf Benutzerdaten zugreift.
- `portletContainer` (`com.infopark.pm.portlet.PortletContainer`)
  Verwaltet die in der Datei `WEB-INF/portlet.xml` definierten Portlets und stellt diese für dynamische Inhalte bereit.
- `documentManager` (`com.infopark.pm.DocumentManager`)
  Liefert die Inhalte für das `ContentServlet`. Die mitgelieferte Implementierung greift dazu auf eine konfigurierbare `com.infopark.pm.DocumentSource` zu.
- `permissionManager` (`com.infopark.pm.PermissionManager`)
  Liefert dem `PermissionFilter` die Namen der Gruppen, die auf auszuliefernde Inhalte zugreifen dürfen. Der `permissionManager` ist ab Version 6.7.1 nicht mehr vorhanden, da der `AuthorizationManager` seine Funktion übernommen hat.
- `authorizationManager` (`com.infopark.pm.user.AuthorizationManager`)
  Überprüft mit Hilfe der angegebenen Methoden (`com.infopark.pm.user.Authorizer`), ob der Benutzer auf einen Inhalt zugreifen darf.
- `templateEngine` (`com.infopark.pm.TemplateEngine`)
  Bereitet dynamische Inhalte so vor, dass das `ContentServlet` das Ergebnisdokument ausrechnen kann.
- `contentHandlerMap` (`java.util.Map`)
  Legt fest, welche Inhalte das `ContentServlet` wie ausliefert (statisch oder dynamisch). Je MIME-Typ kann ein Bean des Typs `com.infopark.pm.doc.ContentHandler` angegeben werden. Für statische und dynamische Inhalte ist jeweils eine Implementierung verfügbar.

4.2.3 Eine Portlet-Web-Applikation konfigurieren

Die in einer Web-Applikation enthaltenen Portlets werden in der Datei `WEB-INF/portlet.xml` konfiguriert. Die Datei hat den folgenden Aufbau:

```
<?xml version="1.0" encoding="UTF-8" ?>
<portlet-app version="1.0"
  xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd
           http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd">
    <portlet>
      ...
    </portlet>
    ...
    <custom-portlet-mode>
      ...
    </custom-portlet-mode>
    ...
    <custom-window-state>
```

```
      ...
    </custom-window-state>
    ...
    <user-attribute>
      ...
    </user-attribute>
    ...
</portlet-app>
```

Die Elemente haben die folgende Bedeutung:

- `portlet` konfiguriert ein Portlet
- `custom-portlet-mode` konfiguriert einen über den Standard hinausgehenden Portlet-Modus wie `about` oder `print`.
- `custom-window-state` konfiguriert einen über den Standard hinausgehenden Fensterstatus.
- `user-attribute` definiert optionale Benutzereigenschaften, die den Portlets zur Verfügung gestellt werden.

Sicherheitseinschränkungen (`security-constraint`) werden vom Infopark Portal Manager nicht unterstützt.

4.2.4 Ein Portlet konfigurieren

Jedes Portlet wird durch ein `portlet`-Element konfiguriert. Im folgenden Beispiel sind die wichtigsten Elemente enthalten:

```
    ...
    <portlet>
        <portlet-name>example</portlet-name>
        <portlet-class>com.infopark.example.Portlet</portlet-class>
        <init-param>
            <description>my init param description</description>
            <name>host</name>
            <value>127.0.0.1</value>
        </init-param>
        <expiration-cache>0</expiration-cache>
        <supports>
            <mime-type>text/html</mime-type>
            <portlet-mode>edit</portlet-mode>
        </supports>
        <supported-locale>en</supported-locale>
        <supported-locale>de</supported-locale>
        <resource-bundle>com.infopark.example.localizer</resource-bundle>
        <portlet-preferences>
            <preference>
                <name>readOnlyPreference</name>
                <value>foo</value>
                <value>bar</value>
                <read-only>true</read-only>
            </preference>
            <preferences-validator>com.infopark.example.Validator</preferences-validator>
        </portlet-preferences>
    </portlet>
    ...
```

Die Elemente haben die folgende Bedeutung:

- `portlet-name`: Die Bezeichnung des Portlets. Der Bezeichner muss innerhalb der Portlet-Web-Applikation eindeutig sein. Über ihn wird das Portlet eingebunden.

- `portlet-class`: Die Java-Klasse, die die Portlet-Funktionalität bereitstellt. Diese Klasse muss in einem jar-Archiv im Verzeichnis `WEB-INF/lib` oder als kompilierte Klasse unter `WEB-INF/classes` enthalten sein.
- `expiration-cache`: Diese Option erlaubt dem Portal Manager, den Portletinhalt für die angegebene Anzahl von Sekunden zu cachen, d.h. in dieser Zeit nicht neu zu berechnen. Bei `0` wird diese Funktion deaktiviert, bei `-1` ist endloses Caching erlaubt (bis eine Aktion ausgeführt wird).
- `init-param`: Ein Konfigurationsparameter für dieses Portlet. Es können beliebig viele `init-param`-Elemente für ein Portlet vorhanden sein.
  - `description`: Optionale Beschreibung
  - `name`: Der Name des Parameters
  - `description`: Der Wert des Parameters
- `supports`: Konfiguriert unterstützte MIME-Typen und Portlet-Modi:
  - `mime-type`: Konfiguriert einen unterstützten MIME-Typ. Je Typ wird ein Element angegeben, der MIME-Typ muss angegeben werden.
  - `portlet-mode`: Weitere vom Portlet unterstützte Portlet-Modi neben `VIEW` können hier konfiguriert werden.
- `supported-locale`: Eine unterstützte Sprache. Typischerweise wird hier nur ein ISO-Kürzel für eine Sprache, optional auch ein Länderkürzel nach einem Unterstrich angegeben. Ein Portlet kann eine oder meherere Sprachen unterstützen, d.h. das Element kann mehrfach angegeben werden.
- `resource-bundle`: Der Name, unter denen die Ressourcen zu finden sind (ohne die Sprache oder Endung `.properties`).
- `portlet-info`: Alternativ zu Ressourcen können Titel, Kurztitel und Schlüsselwörter mit diesem Element auch direkt angegeben werden. Die entsprechenden Unterlemente sind: `title`, `short-title` und `keywords`.
- `portlet-preferences`: Zu jedem Portlet können beliebig viele Voreinstellungen konfiguriert werden. Die Werte dieser Einstellungen können zur Laufzeit geändert werden, sofern die Einstellung nicht als schreibgeschützt gekennzeichnet ist. Optional kann die Überprüfung der Einstellungen mittels einer Validator-Klasse konfiguriert werden.

Die Option `expiration-cache` wird erst ab Version 6.5.2 unterstützt. Für Portlets mit rechenintensiver Implementierung ist diese Option eine Möglichkeit, die Performance des Portals zu verbessern.

4.2.5 Portlet-Modi und Fenster-Status

Ab Version 6.5.1 können Sie eigene, nicht in der JSR168-Spezifikation definierte Portlet-Modi (*engl.* "custom modes") oder Fenster-Status (*engl.* "custom states") einsetzen, indem Sie im `WEB-INF`-Verzeichnis der PM-Web-Applikation die Datei `portlet.xml` anlegen und darin Ihre eigenen Portlet-Modi und Fenster-Status deklarieren.

Beispiel:

```
...
<custom-portlet-mode>
  <portlet-mode>preview</portlet-mode>
</custom-portlet-mode>
<custom-portlet-mode>
  <portlet-mode>urn:javax:portlet:mode:custom:about</portlet-mode>
</custom-portlet-mode>
<custom-window-state>
```

```
      <window-state>solo</window-state>
    </custom-window-state>
    ...
```

4.3 Benutzerverwaltung im Portal

4.3.1 Den Portal Manager mit LDAP oder ADS verwenden

Um den Portal Manager an einen LDAP- oder ADS-Server anzubinden, konfigurieren Sie bitte in der Datei `pm.xml` in der Web-Applikation des Portal Managers das `LdapUserDirectory` bzw. `AdsUserDirectory`-Bean.

Bitte passen Sie die Propertys des `userDirectory` an die kundenspezifischen Werte an. Die einzelnen Parameter sind in der Konfigurationsdatei erläutert.

4.3.2 Benutzereigenschaften in Portlets bereitstellen

Seit Version 6.5.1 können Benutzereigenschaften, die beispielsweise von einem Verzeichnisdienst bereitgestellt wurden, im Portlet über das Requestattribut `javax.portlet.PortletRequest.USER_INFO` ausgelesen werden.

Die vom Portal bereitgestellten Attribute werden in der Datei `portlet.xml` aufgeführt. Das folgende Beispiel zeigt zwei Attribute:

```
<user-attribute>
  <name>user.name.real</name>
</user-attribute>
<user-attribute>
  <name>user.business-info.online.email</name>
</user-attribute>
```

In den meisten Fällen werden diese Attribute jedoch nicht unter den im Standard genannten Namen bereitgestellt. Deshalb bietet der Infopark Portal Manager eine Möglichkeit, Namen aus dem Standard auf bereitgestellte Attributnamen abzubilden. Dazu wird in der Datei `pm.xml` das Property `userAttributeMapper` im Bean `portletContainer` gesetzt.

Das folgende Beispiel zeigt die Verwendung einer einfachen Abbildung der beiden obigen Attribute auf zwei auch vom Content Manager bereitgestellte Attribute:

```
<property name="userAttributeMapper">
  <bean class="com.infopark.pm.user.SimpleAttributeMapper">
    <property name="mapping">
      <value>
        user.business-info.online.email = email
        user.name.real = realName
      </value>
    </property>
  </bean>
</property>
```

Bei dieser Konfiguration kann ein Portlet die Eigenschaft `email` über folgenden Code abfragen:

```
Map userInfo = (Map)request.getAttribute(PortletRequest.USER_INFO);
String email = (String)userInfo.get("user.business-info.online.email");
```

4.3.3 Benutzereigenschaften des Verzeichnisdienstes verwenden

Bei der Anbindung einer LDAP- oder ADS-Benutzerverwaltung können ab Version 6.5.0 des CMS weitere, im Verzeichnisdienst verwaltete Eigenschaften der Benutzer ausgelesen und im Portal verwendet werden. In die Bean-Konfiguration für den Verzeichnisdienst (Datei: `WEB-INF/pm.xml`) werden die ausgelesenen Attribute über das Property `attributes` eingetragen. Beispiel:

```
...
<bean class="com.infopark.pm.user.AdsUserDirectory">
    ...
    <property name="attributes">
     <map>
      <entry key="mail">
       <map>
        <entry key="type"><value>single</value></entry>
       </map>
      </entry>
      <entry key="name">
       <map>
        <entry key="type"><value>single</value></entry>
       </map>
      </entry>
     </map>
    </property>
    ...
</bean>
...
```

Der Wert von `key` in einem Eintrag von `attributes` wird als Attributename der Benutzereigenschaft im Verzeichnisdienst interpretiert.

Für den `type` eines Attributes können die Werte `single` (ein Rückgabewert) sowie `list` (Liste von Rückgabewerten) eingetragen werden.

4.4 Den Portal Manager mit benannten virtuellen Hosts verwenden

Mit Infopark CMS Fiona können mehrere Websites auf benannten virtuellen Hosts (*engl.* "name-based virtual hosts") betrieben werden. Die Websites werden als separate Instanzen mit dem CMS erstellt und gepflegt und vom Portal Manager dynamisch ausgeliefert. Es ist möglich, für mehrere virtuelle Hosts die gleichen Instanzen zu verwenden.

4.4.1 Voraussetzungen und Funktionsweise

Zum Betrieb der virtuellen Hosts werden ein Apache Webserver mit mod_jk, der Trifork Application Server sowie der Infopark Portal Manager benötigt.

Die hereinkommenden Requests für die im Apache konfigurierten virtuellen Hosts sollen zunächst mittels mod_jk zum Trifork-Server weitergeleitet werden. Hierzu dient in der Apache-Konfiguration die Direktive

```
jkMount /* triforkWorker
```

Der Trifork-Server wiederum soll die Requests an die Portal-Manager-Webapplikation weiterleiten, die mittels ihrer Konfiguration einem Hostnamen den entsprechenden Content zuordnet. Hierfür wird das Konfigurationselement `VirtualHostConfig` verwendet.

Da es in der Portal-Manager-Konfiguration nur ein Dokumentenverzeichnis (`documentSource`) für alle virtuellen Hosts gibt, wird dieses auf ein beliebiges Verzeichnis gesetzt, in dem dann für jeden Host ein symbolischer Link angelegt wird, der auf den Content zeigt, der zu dem jeweiligen Host gehört.

4.4.2 Beispielkonfiguration

Apache

```
# /etc/httpd/httpd.conf
...
NameVirtualHost 10.1.110.40:80
NameVirtualHost 10.1.110.40:443
<VirtualHost 10.1.110.40:80>
  ServerName www.infopark.de
  ServerAlias infopark.de
  ErrorLog  /var/log/httpd/error_log
  CustomLog /var/log/httpd/www.infopark.de-access_log combined
  CustomLog /var/log/httpd/www.infopark.de-redir_log redirectlog
  JkMount /* triforkWorker
  ...
</VirtualHost>
<VirtualHost 10.1.110.40:80>
  ServerName www2.iico.de
  ErrorLog  /var/log/httpd/error_log
  CustomLog /var/log/httpd/www.iico.de-access_log combined
  JkMount /* triforkWorker
</VirtualHost>
```

```
# /etc/httpd/workers.properties

worker.list=triforkWorker
# Set properties for triforkWorker (ajp13)
worker.triforkWorker.type=ajp13
worker.triforkWorker.host=localhost
worker.triforkWorker.port=8009
worker.triforkWorker.lbfactor=250
worker.triforkWorker.cachesize=50
worker.triforkWorker.cache_timeout=300
worker.triforkWorker.socket_keepalive=1
worker.triforkWorker.recycle_timeout=300
```

Portal Manager

Für sämtliche virtuellen Hosts wird hier ein einziger Portal Manager verwendet. Wenn man in der Apache-Konfiguration mehrere `workers` definiert, lassen sich die virtuellen Hosts auch auf mehrere Portal Manager aufteilen. Dies könnte aus Gründen der Performance sinnvoll sein, wenn die Portal Manager auf unterschiedlichen Rechnern laufen.

```
# /opt/Infopark/CMS-Fiona/instance/internet/webapps/PM/WEB-INF/pm.xml
...
<bean id="hostConfig" class="com.infopark.pm.VirtualHostConfig">
  <property name="properties">
    <props>
      <prop key="infopark_de.locales">de</prop>
```

```
        <prop key="infopark_de.http.enable">true</prop>
        <prop key="infopark_de.http.host">www2.infopark.de</prop>
        <prop key="infopark_de.http.port">80</prop>
        <prop key="infopark_com.locales">en</prop>
        <prop key="infopark_com.http.enable">true</prop>
        <prop key="infopark_com.http.host">www2.infopark.com</prop>
        <prop key="infopark_com.http.port">80</prop>
        <prop key="iico_de.locales">de</prop>
        <prop key="iico_de.http.enable">true</prop>
        <prop key="iico_de.http.host">www2.iico.de</prop>
        <prop key="iico_de.http.port">80</prop>
      </props>
    </property>
  </bean>
  ...
  <bean id="documentSource" class="com.infopark.pm.doc.WebappDocumentSource">
    <property name="documentSource">
      <bean class="com.infopark.pm.FileDocumentSource">
        <property name="documentRoot" value="/var/www/vhosts" />
        <property name="inputEncoding" value="UTF-8" />
      </bean>
    </property>
    <property name="inputEncoding" value="ISO-8859-1" />
  </bean>
  ...
```

Symbolische Links anlegen

Anschließend werden die symbolischen Links in dem Verzeichnis angelegt, das als `documentSource` angegeben wurde.

```
cd /var/www/vhosts
ln -s /opt/Infopark/CMS-Fiona/instance/internet/export/online/docs infopark_de
ln -s /opt/Infopark/CMS-Fiona/instance/internet/export/online/docs infopark_com
ln -s /opt/Infopark/CMS-Fiona/instance/iico/export/online/docs/iico iico_de
```

Portal Manager deployen

Abschließend muss der Portal Manager neu deployed werden, damit die an seiner Konfiguration vorgenommenen Änderungen wirksam werden. Führen Sie dazu den folgenden Befehl aus dem betreffenden Instanzenverzeichnis aus:

```
./instance/instancename/bin/rc.npsd deploy PM
```

4.5 Entfernte Portlets über WSRP bereitstellen

Ab Version 6.5.1 von Infopark CMS Fiona können mit dem Infopark Portal Manager entfernte Portlets eines WSRP-Producers bereitgestellt werden. Dies geschieht mit Hilfe eines virtuellen Kontext-Pfades, der im Portal Manager angegeben wird. Ein solcher virtueller Kontext-Pfad verweist per URL auf eine WSDL-Datei (WSDL = Web Service Description Language), die beschreibt, wie auf den Service zugegriffen werden kann.

Dadurch, dass entfernte Portlets auf diese Weise eingebunden werden, verhalten sie sich wie Portlets des Infopark Portal Managers und können über die normale `npspm`-Syntax angesprochen werden.

Um WSRP-Portlets bereitzustellen, erweitern Sie bitte das `ProxyPortletContainer`-Bean in der Konfigurationsdatei `webapps/PM/WEB-INF/pm.xml` der betreffenden CMS-Instanz. Fügen Sie bitte das Property `wsrpPathMapping` mit einem Unterelement `value`hinzu. Als Wert des `value`-Elements geben Sie bitte für jeden WSRP-Producer eine Zeile mit dem virtuellen Kontext-Pfad und der dazugehörenden WSDL-URL an. Beispiel:

```
/WS-ORA = http://portalstandards.oracle.com/portletapp/portlets?WSDL
```

Zeigt die URL auf eine WSDL-Datei, die Verweise auf externe Quellen enthält, die aufgrund von Zugriffsbeschränkungen im Netzwerk (Firewall o.ä.) nicht erreichbar sind, so laden Sie die WSDL-Datei herunter und referenzieren Sie sie mit `file:///` über ihren Pfad im lokalen Dateisystem. Beispiel:

```
/WS-IBM = file:///var/temp/portalserver-ibm.wsdl
```

Hier ein exemplarischer Ausschnitt aus der Datei `pm.xml`:

```
<bean id="portletContainer" class="com.infopark.pm.portlet.ProxyPortletContainer">
  <property name="preferencesStorage">
    <bean class="com.infopark.pm.FilesystemPreferencesStorage" />
  </property>
  <property name="cacheManager" ref="cacheManager" />
  <property name="wsrpPathMapping">
    <value>
      /WS-ORA = http://portalstandards.oracle.com/portletapp/portlets?WSDL
      /WS-IBM = file:///var/temp/portalserver-ibm.wsdl
    </value>
  </property>
  <property name="webserviceRefreshInterval" value="0" />
</bean>
```

Die Änderung der Konfiguration muss durch einen Neustart des Application Servers abgeschlossen werden. Funktioniert die Kommunikation des Portal Managers mit dem Producer, können anschließend die Handles der verfügbaren Remote Portlets mit Hilfe der folgenden URL ermittelt werden:

```
http://mein.server:8080/PM/debug/
```

Ein ermitteltes Remote Portlet kann nun analog zu

```
<npspm includeportlet="/WS-ORA/portletHandle" />
```

in den Content eingebunden werden, wobei *`portletHandle`* das Handle des jeweils einzubindenden Portlets ist.

Weitere Propertys

- `webserviceRefreshInterval`: Aktualisierungsintervall für die Liste der Remote Portlets (in Sekunden).

4.6 WSDL-Dateien verfügbar machen

Die WSDL-Datei, die den WSRP-Service beschreibt, enthält normalerweise Verweise auf Schemata oder Namensräume, die auf Webservern im Internet liegen (z.B. `http://www.oasis-open.org/committees/wsrp/specifications/version1/wsrp_v1_bindings.wsdl`, `http://www.w3.org/XML/1998/namespace`). Erlaubt es Ihre eigene Netzwerkkonfiguration nicht, dass der Infopark Portal Manager Verbindungen über HTTP/HTTPS außerhalb des Intranets oder der DMZ aufbaut, so

müssen die Ziele der Verweise auf andere Art verfügbar gemacht werden, damit der Consumer die vollständige Beschreibung der entfernten Dienste erhält. Hierfür gibt es die folgenden Möglichkeiten.

4.6.1 Einsatz eines Proxy-Servers

Wenn es Ihre Netzwerk-Policy erlaubt, dass der Infopark Portal Manager über einen Proxy-Server Verbindungen ins Internet *und* zum Producer aufnimmt, können Sie den Applikationsserver, auf dem der Infopark Portal Manager eingesetzt wird, mit der Java-Option `proxyHost` und `proxyPort` starten.

Wenn Sie den Trifork-Server verwenden, können Sie diese Optionen in die Konfigurationsdatei `rc.npsd.conf` im `bin`-Verzeichnis der betreffenden CMS-Instanz aufnehmen. Beispiel:

```
set conf(triforkArgs) [list server start -devel -vmargs=-server\
  -vmargs=-Xmx256m -vmargs=-Xms256m\
  -vmargs=-XX:MaxPermSize=128m -DproxyHost=mein.server \
  -DproxyPort=3128]
```

Bitte beachten Sie, dass damit *allen Applikationen dieses Applikationsservers der Zugriff ins Internet über den Proxy-Server erlaubt wird*.

4.6.2 Inkludierte Dateien beim Producer lokal bereitstellen

Konfigurieren Sie Ihren Producer so, dass er nur lokale XML-Dateien inkludiert bzw. passen Sie die WSDL-Datei des Producers dahingehend an. Die Referenzen in der WSDL-Datei können dann per Requests an den Producer aufgelöst werden.

4.6.3 Lokale Bereitstellung der WSDL-Datei beim Consumer

Wenn auf der Seite des Producers keine Änderung möglich ist, kann die WSDL-Datei einschließlich aller darin inkludierten XML-Dateien lokal verfügbar gemacht werden. Diese Variante ist am wenigsten flexibel, weil Änderungen beim Producer jedes Mal lokal nachvollzogen werden müssen.

Wählen Sie hierzu bitte über einen Browser die WSDL-URL an und speichern Sie sie lokal ab. Bearbeiten Sie diese Datei, suchen Sie darin alle `import`s zu externen Webservern, laden Sie diese externen XML-Dateien wieder über den Browser und speichern Sie sie lokal ab. Wiederholen Sie dies so lange, bis Sie alle externen inkludierten Dateien lokal abgespeichert haben. Editieren Sie alle gespeicherten Dateien und ändern Sie alle externen Bezüge so, dass sie jeweils auf die lokalen Kopien verweisen. Legen Sie alle Dateien im Applikationsserver an einem Ort ab und konfigurieren Sie in der Datei `pm.xml` das `wsrpPathMapping` für diesen Producer als lokale WSDL-Datei.

5

5 Portlets

Dieser Abschnitt beschreibt die Funktion und Konfiguration von Portlets, die mit dem Infopark Portal Manager verwendet werden können.

Sofern in den Einzelbeschreibungen nichts anderes angegeben ist, ist die Verwendung der in diesem Dokument und seinen Unterdokumenten beschriebenen Portlets – bis auf das Anmeldeportlet – lizenzierungspflichtig.

5.1 News-Portlet

Viele Unternehmen stellen aktuelle Nachrichten als RSS-News-Feeds zur Verfügung. RSS (Really Simple Syndication) ist ein standardisiertes Verfahren, mit dem Informationen verteilt werden können. Ein RSS-Feed ist ein XML-Dokument, das strukturierte Nachrichten enthält (beispielsweise Überschrift, Zusammenfassung, Link), so dass es maschinell verarbeitet werden kann.

Mit dem Infopark CMS können solche Feeds automatisch erzeugt werden, und zwar so, dass bei der Auslieferung durch den Portal Manager die Zugriffsrechte der jeweiligen Portalbenutzer berücksichtigt und die News Feeds damit personalisiert werden können. Es werden alle RSS-Versionen (0.90, 0.91, 0.92, 0.93, 0.94, 1.0 und 2.0) unterstützt.

Für angemeldete Portalbenutzer markiert das Portlet die gelesenen Artikel, die dann in der Übersicht ausgeblendet oder vereinfacht dargestellt werden können. Für Formate RSS 0.93, 0.94 und 2.0 und Atom 0.3 und 1.0 werden dabei ab Version 6.7.0 von Infopark CMS Fiona auch die Veröffentlichungszeitpunkte der Artikel mit den Lesezeitpunkten des Benutzers verglichen, um aktualisierte Artikel gegebenenfalls wieder als ungelesen zu markieren.

Die Verwendung dieses Portlets ist lizenzierungspflichtig.

5.1.1 Bedienung

Das News-Portlet kennt zwei Modi, Ansicht (*view*) und Bearbeitung (*edit*), zwischen denen man mit einem Button in der Kopfleiste umschalten kann. Im Ansichtsmodus zeigt das Portlet die Artikel (*items*) des aktuell ausgewählten Newsfeeds an. Sofern mehrere News-Feeds konfiguriert sind, kann per Umschalter zwischen ihnen gewechselt werden:

Jeder News-Feed besteht wie ein Lesezeichen aus einem Namen und der News-Feed-URL. Im Bearbeitungsdmodus können diese Konfigurationsdaten geändert und News-Feeds umsortiert, hinzugefügt oder gelöscht werden. Hierfür zeigt das Portlet eine Liste der Feeds an:

In den Bearbeitungsmodus kann man nur umschalten, wenn das Portlet mit Rahmen dargestellt wird. Die Bearbeitung des Titels und der URL zeigt der folgende Screenshot:

5.1.2 Konfiguration

Das Portlet hat den Namen `news` und kann in der Datei *portlet.xml* konfiguriert werden. Es gibt drei optionale Initialisierungparameter:

- `updateInterval` definiert die Zeit in Sekunden, nach deren Ablauf ein Newsfeed neu geholt wird.
- `maxItems` definiert die maximale Anzahl von Artikeln pro Benutzer, bei denen das Portlet den Status *gelesen/ungelesen* speichert. Bitte wählen Sie den Wert dieses Parameters mindestens doppelt so groß wie die Anzahl der Einträge in Ihrem Newsfeed. Andernfalls könnte es passieren, dass bereits gelesene Artikel wieder als ungelesen dargestellt werden.
- `maxItemLength` definiert die maximale Textlänge der Beschreibung eines Artikels. Ist die Beschreibung länger, wird sie passend abgeschnitten und '...' an sie angehängt.

Beispiel für die Initialisierungsparameter

```
<init-param>
  <name>updateInterval</name>
  <value>5</value>
</init-param>
<init-param>
  <name>maxItems</name>
  <value>5</value>
</init-param>
<init-param>
  <name>maxItemLength</name>
  <value>160</value>
</init-param>
```

Darüber hinaus gibt es eine Einstellung, `feeds`, mit der beliebig viele News-Feeds als Vorgabe für neue Benutzer vorkonfiguriert werden können. Jede Feed-Definition besteht aus dem Namen und der URL des Feeds, getrennt durch das Zeichen | (Name und URL dürfen dieses Zeichen also nicht enthalten.) Beispiel:

```
<portlet-preferences>
```

```
  <preference>
    <name>feeds</name>
    <value>
      Infopark News|http://www.infopark.de/rss
    </value>
    <value>
      Slashdot News|http://slashdot.org/index.rss
    </value>
    <value>
      Yahoo! News|http://rss.news.yahoo.com/rss/topstories
    </value>
  </preference>
</portlet-preferences>
```

5.1.3 Verwendung

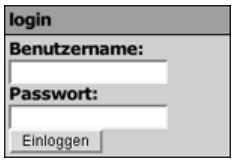
Das Portlet wird mit

```
<npspm includePortlet="/PM-PL/news" ... />
```

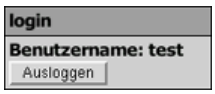
in Layoutdateien eingebunden. Wenn das Portlet sich in einer anderen Web-Applikation befindet, geben Sie deren Namen anstelle von `PM-PL` an.

5.2 Anmelde-Portlet

Das Anmelde-Portlet bietet dem Portal-Benutzer die Möglichkeit, sich anzumelden:



Ferner erlaubt das Portlet angemeldeten Benutzern, sich abzumelden:



5.2.1 Konfiguration

Die Benutzer können sich mit Hilfe des in der Datei `/instance/instName/webapps/PM/WEB-INF/pm.xml` konfigurierten Benutzermanagers authentifizieren.

5.2.2 Verwendung

Das Portlet wird mit

```
<npspm includePortlet="/PM-PL/login" ... />
```

in Layoutdateien eingebunden.

5.3 Portlet zur Bearbeitung der aktuellen Seite

Das Edit-page-Portlet bietet dem Benutzer die Möglichkeit, das aktuell im Portal dargestellte Dokument im Redaktionssystem, also im CMS Navigator zu bearbeiten.

Die Verwendung dieses Portlets ist lizenzierungspflichtig.

5.3.1 Bedienung

Klickt man auf den Link im Portlet, so öffnet sich eine CMS-Navigator-Seite mit der entsprechenden selektierten Datei in einem neuen Browserfenster. Sofern der Benutzer am Portal angemeldet ist, versucht das Portlet, ihn mit dem selben Benutzernamen und Passwort am Navigator anzumelden. Gelingt dies nicht, wird zunächst die Anmeldeseite angezeigt.

Dieses Portlet ist insbesondere nützlich, wenn die Portalbenutzer gleichzeitig als CMS-Redakteure tätig sind.

5.3.2 Konfiguration

Das Layout dieses Portlets (wie auch aller anderen) wird mit Hilfe von Velocity-Templates festgelegt, die in der Web-Applikation `PM` unter `/WEB-INF/templates` liegen. Das Layout kann beliebig angepasst werden.

5.3.3 Verwendung

Das Portlet wird mit

```
<npspm includePortlet="/PM-IP/editPage" ... />
```

in Layoutdateien eingebunden.

5.4 Formular-Portlet

Mit dem Formular-Portlet lässt sich auf einfache Weise eine Abfolge von Formularseiten bereitstellen.

Die Verwendung dieses Portlets ist lizenzierungspflichtig.

Mit entsprechenden Buttons kann der Benutzer zur jeweils vorigen oder nächsten Seite der Formular-Abfolge wechseln. Wird das letzte Formular "abgeschickt", so wird üblicherweise eine Aktion ausgeführt. Beispielsweise können die Formulardaten per E-Mail verschickt werden.

Die Masken und ihre Abfolge sowie die auszuführende Aktion am Ende werden in einer XML-Datei definiert. Mit Hilfe von Velocity-Templates wird die dynamische Darstellung der einzelnen Formulare auf der Webseite gesteuert.

Jede Abfolge wird in einem Verzeichnis unterhalb von `WEB-INF/templates/flow` definiert. Das Portlet kann unter Angabe des Namens dieses Verzeichnisses in den Content eingebunden werden. Diese Definition umfasst immer eine XML-Datei `flow.xml`, die die Formularseiten und deren Felder festlegt. Zu jeder Formularseite wird ein in der Definition referenziertes Velocity-Template zur Darstellung der HTML-Seite in dem Verzeichnis abgelegt. Die Definition umfasst ferner die oben erwähnte Aktion. Diese Aktion wird durch eine Java-Klasse implementiert. Als typischer Anwendungsfall wird der Versand der Formulardaten per E-Mail bereitgestellt.

5.4.1 Konfiguration

Formulardefinitionen

Die Formularseiten werden in der Datei `flow.xml` definiert. Das Hauptelement in dieser Datei ist stets `flow`. Dieses Element kann ein Attribut `result` haben, das das Ergebnistemplate definiert. Wenn `result` nicht angegeben wurde, wird `result.vm` verwendet. Unterhalb von `flow` wird jede Formularseite mit einem `form`-Element definiert sowie die abschließende Aktion mit einem `action`-Element angegeben.

Jedes `form`-Element definiert genau eine Formularseite. Mit seinem Attribut `template` wird der Name des Velocity-Templates angegeben, das zur Darstellung dieser Formularseite verwendet wird. Diese Template-Datei muss im selben Ordner wie die Definitionsdatei `flow.xml` liegen. Hier ein Beispiel:

```
<?xml version="1.0" encoding="UTF-8" ?>
<flow>
  <form template="contact.vm">
    <field name="gender" type="select"
      required="true" flags="menu">
      <item default="true">female</item>
      <item>male</item>
    </field>
    <field name="firstName" type="text"
      regex="([\p{Lu}][\p{Ll}.-]* *)+" error="errorInvalidName" />
    <field name="lastName" type="text" required="true"
      regex="([-\p{Lu}\p{Ll}]+ *)+" error="errorInvalidName" />
    <field name="email" type="text"
      required="true" validator="email" />
    <field name="subject" type="text" required="true" />
    <field name="message" type="text"
      required="true" flags="area" />
  </form>
    <action name="sendEmail">
    <param name="receiver">playland@infopark.de</param>
    <param name="template">email.vm</param>
    <param name="senderField">email</param>
    <param name="subjectField">subject</param>
  </action>
</flow>
```

Ein `form`-Element kann beliebig viele `field`-Elemente haben. Jedes `field`-Element repräsentiert ein Feld der betreffenden Formularseite. `field`-Elemente können die folgenden Attribute haben:

- `name`: Der auf der Seite eindeutige Name des Feldes.
- `type`: Der Typ des Feldes. Erlaubte Typen sind in der Datei `WEB-INF/templates/flow.properties` definiert:
  - `text`: einfacher Text
  - `select`: Auswahl
  - `multiselect`: Mehrfachauswahl

  - `file`: eine hochzuladende Datei
  - `boolean`: Wahrheitswert
  - `date`: Datum
  - `month`: Monat
- `required`: gibt an, ob das Feld ein Pflichtfeld ist.
- `flags`: eine kommaseparierte Liste von Darstellungshinweisen
- `value`: eine Vorbelegung (beim Typ `text`)
- `validator`: ein Test des Wertes. Mögliche Tests sind in der Datei `WEB-INF/flow.properties` definiert.
- `regex`: ein Test des Wertes über einen regulären Ausdruck (für den Typ `text`)
- `error`: der Fehler, der beim Fehlschlagen des `regex`-Tests angegeben wird, andernfalls `errorInvalid`

Die Attribute `name` und `type` sind obligatorisch. Es kann nur entweder das Attribut `validator` oder `regex` angegeben werden, nicht beide. Bei den Typen `select` und `multiselect` können die Werte, aus denen der Benutzer einen bzw. mehrere auswählen kann, mit Hilfe von `item`-Elementen im `field`-Element angegeben werden. Der bereits ausgewählte Wert (bei `select`) bzw. die bereits ausgewählten Werte (bei `multiselect`) können mit dem Attribut `default="true"` des jeweiligen `item`-Elements definiert werden.

Hat das `item`-Element ein Attribut `value`, so wird dessen Wert als Feldwert verwendet, wenn der Benutzer das Element ausgewählt hat, während der Inhalt des Elements zur Darstellung dient. Ist `value` nicht angegeben, wird zur Darstellung der lokalisierte Inhalt des `item`-Elements verwendet, während der Inhalt selbst der eigentliche Feldwert ist.

Die Aktion am Ende der Formularabfolge wird über ein `action`-Element definiert. Über dessen Pflichtattribut `name` wird die Aktion bestimmt; die erlaubten Namen sind wiederum in der Datei `WEB-INF/flow.properties` definiert. Wie bei den Tests wird auch hier eine Java-Klasse referenziert, die die Aktion implementiert. Zu übergebende Argumente können in `param`-Elementen innerhalb von `action` angegeben werden. Jedes `param`-Element hat über das Pflichtattribut `name` einen eindeutigen Namen. Der Inhalt eines solchen Elements ist der Argumentwert.

Darstellung der Formulare mit Velocity-Templates

Jede Formularseite wird über ein Velocity-Template dargestellt. Im Kontext stehen folgende Variablen für den Zugriff auf dynamische Inhalte bereit:

- `$fields`: alle Felder der Formularseite über ihren Namen
- `$errors`: alle aufgetretenen Fehler als Liste
- `$text`: Lokalisierungen (s. u.)
- `$page`: die Nummer der aktuellen Seite beginnend mit 1
- `$pages`: die Anzahl der Seiten
- `$action`: die URL für das Formular

Für die Darstellung der Felder stehen mehrere Makros bereit, die aus der Datei `WEB-INF/templates/flow/macros.vm` stammen:

- `#renderLabel`: stellt einen lokalisierten Feldtitel dar
- `#renderField`: stellt das Feld entsprechend des Typs und der Flags (s. u.) dar
- `#renderButtons`: stellt eine Liste von Buttons dar

- `#renderErrors`: stellt alle aufgetretenen Fehler dar

Die Felddarstellung kann durch Flags folgendermaßen beeinflusst werden:

- `area` (`text`): mehrzeilige Texteingabe
- `password` (`text`): nicht lesbare Texteingabe
- `sorted` (`select`, `multiselect`): Einträge alphabetisch sortierten
- `menu` (`select`, `multiselect`): Auswahl als Dropdown-Menü

Hier ein Beispiel-Template, das ein Kontaktformular darstellt:

```
<form action="$action" method="post" class="contact">
  #renderErrors()
  <table cellpadding="0" cellspacing="0" border="0">
    <tr>
      <td class="contact">#renderLabel($fields.gender)
        <div>#renderSelectField($fields.gender)</div>
      </td>
      <td class="contact">#renderLabel($fields.lastName)<br>
        #renderTextField($fields.lastName)
      </td>
      <td class="contact">#renderLabel($fields.firstName)<br>
        #renderTextField($fields.firstName)
      </td>
    </tr>
    <tr>
      <td class="contact" colspan="3">
        #renderLabel($fields.email)<br>
        #renderTextField($fields.email)
      </td>
    </tr>
    <tr>
      <td class="contact" colspan="3">
        #renderLabel($fields.subject)<br>
        #renderTextField($fields.subject)
      </td>
    </tr>
    <tr>
      <td class="contact" colspan="3">
        #renderLabel($fields.message)<br>
        #renderTextField($fields.message)
      </td>
    </tr>
    <tr>
      <td class="contact" colspan="3">
        #renderButtons(["Ok"])
      </td>
    </tr>
  </table>
</form>
```

Beachten Sie bitte auch die Beschreibung der Velocity-Syntax.

Lokalisierung

Im Ordner der Formularseitendefinition wird die Lokalisierung für Feldtitel, Schalter, Fehler und weitere Texte hinterlegt. Auf die lokalisierten Zeichenketten kann man im Template über die Kontextvariable `text` zugreifen. Die Lokalisierungen werden im Verzeichnis als "Java-Resourcebundle" mit dem Namen `localizer` abgelegt. Es besteht aus einer Datei je Sprache. Diese Datei hat den Namen `localizer_`*`Language`*`.properties`, wobei *`Language`* für ein aus zwei Zeichen bestehendes Sprachkürzel wie `de` oder `en` steht.

Schalter haben üblicherweise das Präfix `button`, Fehler das Präfix `error`, Feldtitel den Namen des Feldes. Es können auch zusätzliche Meldungen, beispielsweise für das Aktionsergebnis, hinzugefügt werden.

5.4.2 Verwendung

Das Portlet wird mit

```
<npspm includePortlet="flow" instance="flowname" />
```

in den Content eingebunden, wobei `flowname` für den Namen des Verzeichnisses steht, in dem die Formularabfolge definiert ist.

Wenn das Portlet ausgeführt wird, stellt es die erste Formularseite im Portlet dar. Für die Navigation werden Schalter verwendet. Diese werden typischerweise mit `#renderButtons` dargestellt. Die Bedeutung der Schalter mit den folgenden Namen ist festgelegt:

- `Cancel`: bricht den Formularablauf ab, löscht alle Daten und kehrt zur ersten Seite zurück.
- `Back`: springt zur vorigen Seite, sofern diese vorhanden ist. Eingegebene Daten bleiben erhalten.

Alle anderen Schalter haben die Funktion von *Weiter* oder *OK*. Wird ein solcher Schalter auf der letzten Formularseite gedrückt, so wird die definierte Aktion ausgeführt. Tritt dabei kein Fehler auf, so wird das Ergebnis mit Hilfe des Ergebnistemplates dargestellt.

Verursacht ein Feld einen Fehler oder tritt bei der Aktion ein Fehler auf, so wird die aktuelle Seite nicht verlassen, sondern mit entsprechenden Fehlern erneut dargestellt. Die Fehler sollten an geeigneter Stelle durch das Makro `#renderErrors` dargestellt werden.

Beispiel: E-Mail-Versand

Als typischer Anwendungsfall ist eine Aktion zum Versand der Formulardaten als E-Mail bereits implementiert. Die Aktion `sendEmail` kann an unterschiedliche Anforderungen angepasst werden. Dazu stehen Argumente im `action`-Element zur Verfügung (siehe die Beispieldefinition oben):

- `receiver`: der oder die Empfänger der E-Mail. Sollte dieses Argument fehlen, so wird die E-Mail an den Absender verschickt.
- `template`: der Name des E-Mail-Templates
- `sender` /`senderField`: der Absender bzw. das Feld, aus dem der Absender gelesen wird. Achtung, der Absender muss vom Mailserver akzeptiert werden!
- `subject` / `subjectField`: die Betreffzeile bzw. das Feld, aus dem die Betreffzeile gelesen wird.

Beispiel: E-Mail-Versand mit Bestätigungsnachricht

Ergänzend zur Aktion `sendEmail` ist die Aktion `sendEmailAndConfirmation` implementiert. Mit ihr kann zusätzlich zur eigentlichen Nachricht auch eine Bestätigungsnachricht versandt werden.

Zusätzlich zu den Parametern der Aktion `sendEmail` stehen die folgenden Argumente im `action`-Element zur Verfügung:

- `confirmationReceiverField`: das Feld, aus dem der Empfänger der Bestätigungsnachricht gelesen wird.
- `confirmationTemplate`: der Name des Templates für die Bestätigungsnachricht

- `confirmationSubjectField`: das Feld, aus dem die Betreffzeile für die Bestätigungsnachricht gelesen wird.

Die Feldwerte werden typischerweise mit den Makros `#renderValue` und (ab Version 6.7.2) `#renderHtmlEscapedValue` in das Template übernommen.

5.5 Such-Portlet

Mit Hilfe des Such-Portlets lassen sich Suchformulare und Trefferlisten in eine Website integrieren.

Sowohl die Funktion als auch das Layout der Suche können angepasst werden, ohne dass Java-Code geändert zu werden braucht. Das Portlet kann sowohl Anfragen an den Content Management Server als auch an den Search Server senden. Es verwendet die in der Konfigurationsdatei `pm.xml` im dem Eintrag `searchEngine` konfigurierte Suchmaschine.

Jede Suche wird in der Portlet-Web-Applikation in einem Verzeichnis unterhalb des instanzenspezifischen Web-Applikationsverzeichnisses `WEB-INF/templates/search` definiert.

5.5.1 Verwendung

Das Such-Portlet wird folgendermaßen in Layoutdateien eingebunden:

```
<npspm includeportlet="/PM-PL/search" instance="dealer" language="de" />
```

Der Instanzenname `dealer` entspricht hierbei dem Namen des Verzeichnisses, das die Definition der Suche enthält. Mit `language` kann optional die vom Portlet zu verwendende Sprache angegeben werden. Fehlt diese Angabe, wird die im Browser eingestellte Sprache verwendet.

5.5.2 Konfiguration

Definition einer Suche

Eine Suche wird mit Hilfe zweier Velocity-Templates definiert, die sich im oben angegebenen Verzeichnis unterhalb der jeweiligen Web-Applikation befinden müssen. Die Datei `view.vm` legt das Layout des Suchformulars und der Trefferliste fest. Mit Hilfe der Datei `config.vm` wird die vom Portlet auszuführende Suchanfrage definiert.

Da das Portlet sich mehrsprachig darstellen kann, wird je Sprache bzw. Locale der Verity-Suchmaschine eine Lokalisierungsdatei mit dem Namen `localizer_`*`xy`*`.properties` benötigt, wobei *`xy`* im obigen Dateinamen durch das jeweilige Sprachkürzel ersetzt werden muss. Diese Dateien enthalten benannte Zeichenketten in der folgenden Form:

```
title: Händler
buttonSearch: Suchen
errorNoResults: Ihre Suchanfrage hat keine Ergebnisse geliefert.
```

Beachten Sie bitte, dass die Kodierung der Localizer-Dateien UTF-8 sein muss.

Aufbau der Konfigurationsdatei `config.vm`

Nachdem das vom Portlet generierte Suchformular abgeschickt wurde, lädt und evaluiert das Portlet das Velocity-Template `config.vm`, um aus den Eingaben im Formular die Suchanfrage zu erzeugen. Die Parameter aus dem Suchformular stehen im Template unter ihrem Namen zur Verfügung.

Die Auswertung des Templates ergibt ein XML-Dokument, aus dem das Portlet ein Suchanfrage-XML-Dokument erzeugt, das der Search Server verarbeiten kann. Das Portlet sendet dieses Dokument an den Search Server und erhält als Antwort ein XML-Dokument, das die anzuzeigenden Suchergebnisse enthält.

Das vom Velocity-Template erzeugte XML-Dokument muss die folgende Struktur haben:

```
<query>
  <condition>...</condition>
  <result-fields>...</result-fields>
  <sort>...</sort>
  <collections>...</collections>
  <page-size>...</page-size>
  <max-hits>...</max-hits>
  <min-score>...</min-score>
</query>
```

Mit Ausnahme der `condition` sind alle Elemente optional. Hier als Beispiel die Datei `config.vm` für eine Suche nach Dokumenten, die einen Suchbegriff enthalten, und die für jeden Benutzer frei zugänglich sind:

```
<query>
  <condition>
    <and>
      <vql-statement>
        &lt;#MANY&gt;&lt;#STEM&gt;${searchText}
      </vql-statement>
      <vql-statement>
        &lt;#MANY&gt;&lt;#STEM&gt;free &lt;#IN&gt;
        noPermissionLiveServerRead
      </vql-statement>
    </and>
  </condition>
  <result-fields>
    <field>title</field>
    <field>name</field>
    <field>visiblePath</field>
    <field>ip_abstract</field>
  </result-fields>
  <sort>
    <criterion>
      <field>pl_PLZ</field>
      <ascending/>
    </criterion>
    <criterion>
      <field>name</field>
      <ascending/>
    </criterion>
  </sort>
  <page-size>5</page-size>
  <collections>
    <collection>cm-contents-${language}</collection>
  </collections>
  <log>
    <context>search</context>
#if ($user.isLoggedIn())
    <login>${user.login}</login>
#end
    <query>${searchText}</query>
```

```
  </log>
</query>
```

Die Elemente der Suchanfrage haben die im Folgenden erläuterte Bedeutung:

- `condition`: Dies legt die Suchanfrage fest. Die `condition` kann die folgenden Operatoren enthalten:
  - `and`: dieses Element verknüpft die darin enthaltenen Elemente logisch mit *und*.
  - `contains`: sucht nach Dateien, bei denen das Feld `field` die Zeichenkette `value` enthält. Beispiel:

```
<contains>
  <field>keywords</field>
  <value>Dienstleistung</value>
</contains>
```

  - `contains-match`: sucht nach Dateien, bei denen das Wildcard-Muster in `value` auf den Wert des Feldes `field` passt. Erlaubte Wildcards sind Stern (`*`) und Fragezeichen (`?`).
  - `equals`: sucht nach Dateien, bei denen das Feld `field` exakt dem Wert `value` entspricht. Beispiel:

```
<equals>
  <field>name</field>
  <value>mastertemplate</value>
</equals>
```

  - `or`: verknüpft die im Element enthaltenen Elemente logisch mit *oder*.
  - `starts-with`: sucht nach Dateien, bei denen das Feld `field` mit der Zeichenkette `value` beginnt.
  - `vql-statement`: enthält die Suchanfrage in der expliziten Verity-Anfrage-Sprache. Details zur Syntax finden Sie im Handbuch zum Infopark Search Server. Beachten Sie bitte, dass Sonderzeichen als HTML-Entities angegeben werden müssen, also beispielsweise `<` als `<`.
- `result-fields`: Dieses Element spezifiziert in seinen `field`-Unterelementen die Felder der gefundenen Dokumente, die ins Suchergebnis aufgenommen werden sollen, damit sie auf den Ergebnisseiten angezeigt werden können.
- `sort`: Hier lassen sich ein oder mehrere Kriterien spezifizieren, nach denen die Treffer sortiert werden sollen. Jedes Kriterium besteht aus dem Feldnamen `field`, dessen Inhalt zur Sortierung herangezogen wird, sowie optional einem der Elemente `ascending` oder `descending`, um die Sortierreihenfolge zu bestimmen.
- `page-size`: Mit diesem Element kann die Anzahl der Treffer angegeben werden, die das Portlet je Ergebnisseite anzeigen soll.
- `collections`: Geben Sie hier die zu durchsuchenden Collections an. Wird diese Angabe weggelassen, werden alle Collections durchsucht.
- `log`: Geben Sie hier die zu protokollierenden Einträge an. Mögliche Einträge sind `context`, `login` und `query`. Der Eintrag `context` muss angegeben sein, andernfalls findet keine Protokollierung statt.

Bitte beachten Sie, dass die zu durchsuchenden und auch die von der Suchmaschine zurückzugebenden Felder (siehe oben `result-fields`, `sort`) in der Verity-Konfiguration definiert worden sein

müssen, bevor die betreffende Collection erzeugt wird (siehe die Dokumentation zur Infopark Search Cartridge).

Aufbau der Konfigurationsdatei view.vm

Mit diesem Velocity-Template wird der HTML-Text des Suchformulars und der Trefferliste generiert.

Velocity-Kontext zugreifen:</p>

- `$action`: Die im Suchformular zu verwendende Aktion.
- `$params`: Liste der bisherigen Suchparameter. Interne Parameter (deren Name beginnt mit einem Unterstrich) tauchen in dieser Liste nicht auf.
- `$text`: bietet Zugriff auf lokalisierte Texte (aus den Lokalisierungsdateien im Definitionsverzeichnis).
- `$locale`: das aktuelle Locale.
- `$search`: Tool, mit dem URLs erzeugt werden können. Die folgenden Methoden sind verfügbar:
  - `String getPageUrl(Page page)`: liefert eine URL, mit der auf die angegebene Seite der Trefferliste gesprungen werden kann.
  - `String getSearchUrl(String parameter, String value)`: liefert eine URL, mit der eine Suche mit dem angegebenen Parameter durchgeführt werden kann.
  - `String highlight(String word, String text, String prefix, String suffix)`: Klammert jedes Vorkommen des angegebenen Wortes `word` im Text `text` in die Zeichenketten `prefix` und `suffix` ein.
- `$result`: bietet mit den folgenden Methoden Zugriff auf die Trefferliste:
  - `int getHitCount()`: liefert die Anzahl der Treffer.
  - `List getPages()`: liefert die Liste der `Page`-Objekte im Suchergebnis.
  - `Page getCurrentPage()`: liefert das `Page`-Objekt, das die aktuelle Seite der Trefferliste repräsentiert.

Die Eigenschaften der oben aufgeführten Objekte sind in der mit dem Portlet gelieferten javadoc-Dokumentation beschrieben. Beispiel (oben das Formular, unten das Verlocity-Template):

```
<form method="get" action="$action">
  <input type="text" name="searchText" value="$!params.searchText"/>
  <input type="submit" name="_buttonSearch"
    value="$text.buttonSearch"/>
</form>

#if ($result)
  #if ($result.hitCount == 0)
    $text.errorNoResults
  #else
    <ul>
    #foreach ($item in $result.currentPage.hits)
      #set($path = $document.getUrl($item.visiblePath))
      <li><a href="$path">$item.title</a></li>
    #end
    </ul>
  #end
#end
```

Den Zugriff auf die Suchmaschine konfigurieren

Die Suchmaschine wird in der Datei `pm.xml` als bean `searchEngine` konfiguriert. Es stehen zwei Implementierungen zur Auswahl, der direkte Zugriff über den Search Server und der Zugriff über den Content Management Server.

`SesSearchEngine`: Für die Suche auf dem Live-System werden die Suchanfragen direkt an den dort laufenden Search Engine Server geschickt:

```
<bean id="searchEngine"
    class="com.infopark.libs.search.ses.SesSearchEngine">
  <property name="host"><value>localhost</value></property>
  <property name="port"><value>3011</value></property>
</bean>
```

`AdvancedCmSearchEngine`: Bei der Suche über den Content Management Server wird das Redaktionssystem durchsucht. Der Content Management Server leitet die Anfragen an den für das Redaktionssystem zuständigen Search Engine Server weiter, fügt zu jedem Treffer jedoch den Pfad der gefundenen CMS-Datei hinzu:

```
<bean id="searchEngine"
    class="com.infopark.libs.search.cm.AdvancedCmSearchEngine">
  <property name="host"><value>localhost</value></property>
  <property name="port"><value>3002</value></property>
  <property name="user"><value>root</value></property>
  <property name="tokenManager">
    <ref bean="tokenManager"/>
  </property>
</bean>
```

5.5.3 Beispiel

Das folgende Beispiel zeigt, wie eine neue Suche im Democontent-Portal erstellt wird. Ein einzugebender Suchbegriff soll im Hauptinhalt und im Titel sowie in der Zusammenfassung gesucht werden, also in den Feldern mit den Namen `blob`, `title` und `ip_abstract`. Auf den Ergebnisseiten sollen die Felder `title` und `ip_abstract` ausgegeben werden. Die Zusammenfassung soll nur angezeigt werden, wenn sie Text enthält, also nicht leer ist.

Vorbereitung der Suchmaschinen-Indizes

Die oben genannten Felder können erst verwendet werden, nachdem die Konfiguration der Suchmaschine auf der Live-Seite so erweitert wurde, dass die zusätzlichen Felder zur Verfügung stehen und beim Indizieren der Dokumente gefüllt werden.

Öffnen Sie hierfür die zu der Instanz und der betreffenden (zu durchsuchenden) Collection gehörende Datei

```
  instance/instanceName/config/vdk/styles/collectionName/style.ufl
```

und tragen Sie zuätzlich zu den bestehenden Feldern die folgenden Felder ein:

```
# ----------------------------------------------------------------
# Specify additional application-specific fields here in their own
# data-table[s].
```

```
data-table: nps
{
  **** Bisherige Felder stehen hier
  ****
  **** Neue Felder

  varwidth:   ip_abstract dxa
  autoval:    collection DBNAME
}
```

Stoppen Sie vor den nachfolgenden Schritten den Search Engine Server:

```
./rc.npsd stop SES
```

Damit die Änderungen wirksam werden, muss die Collection neu angelegt und die Dokumente müssen neu indiziert werden. Starten Sie hierfür den Search Engine Server im Singlemodus:

```
./SES -single
```

Führen Sie nun die folgenden Schritte für die betreffende Collection aus:

```
% deleteCollection collectionName
% createCollection collectionName
% exit
```

Starten Sie den Search Engine Server:

```
./rc.npsd start SES
```

Melden Sie sich nun am Content Management Server an:

```
./client localhost 3001 root demo
```

Indizieren Sie mit dem folgenden Kommando alle NPS-Dateien neu:

```
CM>indexAllObjects
CM>exit
```

Die neue Suche erstellen

Wechseln Sie in das Verzeichnis

```
/instance/instanceName/webapps/PM-PL/WEB-INF/templates/search
```

und legen Sie unter dem Namen `bodySubjects` eine Kopie des vorhandenen Verzeichnisses `nameTitle` an. Wechseln Sie nun in den Ordner und bearbeiten Sie die beiden Dateien `config.vm` und `view.vm`.

In der Datei `config.vm` wird die Suchanfrage konfiguriert. Ändern Sie zu diesem Zweck den Inhalt des `condition`-Elements so, dass er nur das folgende `vql-statement`-Element enthält:

```
<vql-statement>
  (&amp;quot;${suche}&amp;quot; &amp;lt;#IN&amp;gt; blob) &amp;lt;#OR&amp;gt;
  (title &amp;lt;#SUBSTRING&amp;gt; &amp;quot;${suche}&amp;quot;) &amp;lt;#OR&amp;gt;
  (ip_abstract &amp;lt;#SUBSTRING&amp;gt; &amp;quot;${suche}&amp;quot;)
</vql-statement>
```

Beachten Sie, dass die Suchanfrage aus mehreren Teilen (eines pro Feld) besteht, die miteinander mit OR verknüpft sind. Da der `blob` (der Hauptinhalt) eine Dokumentzone und kein Feld ist (wie `title` und `ip_abstract`), wird für ihn der Operator IN verwendet, der Dokumentzonen durchsucht. Der Operator SUBSTRING dagegen durchsucht Feldinhalte.

Da in den Suchergebnissen die Inhalte des Feldes `ip_abstract` vorkommen sollen, muss das Element `result-fields` um dieses Feld ergänzt werden:

```
<result-fields>
  <field>objId</field>
  <field>name</field>
  <field>title</field>
  <field>score</field>
  <field>visiblePath</field>
  <field>ip_abstract</field>
</result-fields>
```

Die Such- und Ergebnisseite erstellen

Die Such- und Ergebnisseite werden in der Datei `view.vm` zusammengestellt. Der obere Teil definiert das Suchformular:

```
<form action="$action" method="post">
  <div>
    Ihre Suche <input type="text" name="suche"
        value="$!params.suche" />
  </div>
  <input type="submit" name="dialog.buttonOk"
      value="$text.buttonOk" />
</form>
```

Der untere Teil der Datei `view.vm` sorgt dafür, dass die Ergebnisse angezeigt werden, nachdem der Benutzer das Suchformular abgeschickt hat. Alle Ergebnisse sollen als Liste dargestellt werden, die je Treffer dessen Titel (verlinkt) und die Zusammenfassung enthält, also in folgender Form:

```
<ul>
  <li>Verlinkter Titel<br />
  ip_abstract</li>
  <li>...</li>
  <li>...</li>
</ul>
```

Um diese Darstellung zu erreichen, wird der folgende Code verwendet:

```
#if ($result)
  #if ($result.hitCount == 0)
    <div>$text.noHits</div>
  #else
    <!-- Anzahl der Ergebnisse -->
    <div>$text.hitCount $result.hitCount</div>
    <!-- Beginn Auflistung der Ergebnisse -->
    <ul>
```

```
      #foreach ($item in $result.currentPage.hits)
        <li>
          #set($path = $document.getUrl($item.visiblePath))
          #if ($path)<a href="$path">#end
          $item.title
          #if ($path)</a>#end
          #if ($item.ip_abstract)<br />$itm.ip_abstract#end
        </li>
     #end
    </ul>
    <!-- Ende Auflistung der Ergebnisse -->
    <!-- Links zu anderen Ergebnisseiten erzeugen -->
    <div>
    #foreach ($page in $result.pages)
      #if ($page.isCurrent())
        $page.number
      #else
        <a href="$search.getPageUrl($page)">$page.number</a>
      #end
    #end
    </div>
    <div>$text.page $result.currentPage.number $text.pageOf
        $result.pages.size()
    </div>
    <!-- Ende Übersicht Ergebnisseiten -->
  #end
#end
```

Um die Pfade in der Ergebnisliste vom Anfang her um ein Verzeichnis zu kürzen, so dass ein übergeordneter Ordner nicht angezeigt wird, verwenden Sie in der `foreach`-Schleife den folgenden Code:

```
## Verzeichnistrenner (slash) suchen
#set( prefix $path.indexOf("/", 1) )
## Statt mit $path folgendermaßen auf den Pfad zugreifen
$path.substring($prefix + 1)
```

Das Portlet in eine Webseite einbinden

Um das Portlet in eine Webseite einzubinden, fügen Sie bitte die folgende Zeile in eine Layoutdatei ein, mit der anzuzeigende Seiten erzeugt werden:

```
<npspm includePortlet="/PM-PL/search" instance="bodySubjects" />
```

In der separaten Vorschau wird nun das Portlet angezeigt und kann verwendet werden.

Eingabefelder des Suchformulars leeren

Das Suchportlet speichert die in seinem Suchformular enthaltenen Eingaben. Ruft ein Website-Besucher nach einer Suche erneut die Suchseite auf (die Seite, die das Suchportlet enthält), so enthält das Formular die letzten Eingaben, auch wenn die Suchseite zwischenzeitlich verlassen wurde.

Da dieses Verhalten normalerweise nicht erwünscht ist, kann man das Portlet die Eingabefelder leeren lassen. Bis Version 6.0.x fügen Sie hierzu in die URL auf die Suchseite den Parameter `reset=true` hinzu. Beispiel:

```
<a href="/suchen.html?reset=true">Suche</a>
```

Ab Version 6.5.1 fügen Sie stattdessen einen Link auf das Remote-Control-Portlet ein. Beispiel:

```
<npspm includePortlet="/PM-PL/remoteControl" instance="search">
targetUrl=/suchen.html
emptySearch=true
emptySearchLinkText=Zur Suche
</npspm>
```

Der verlinkte Text ist im Portlet selbst konfiguriert und lautet "Neue Suche". Wenn Sie den verlinkten Text wie im obigen Beispiel mit dem Parameter `emptySearchLinkText` beim Portlet-Aufruf angeben können möchten, ersetzen Sie bitte in der zum Control-Portlet gehörenden Template-Datei `webapps/PM-PL/WEB-INF/templates/remote/search/view.vm` die Zeile

```
  <a href="javascript:document.emptySearch.submit();">$text.emptySearch</a>
```

durch folgenden Code:

```
#if ($params.emptySearchLinkText)
    #set ($linkText = $params.emptySearchLinkText)
#else
    #set ($linkText = $text.emptySearch)
#end
    <a href="javascript:document.emptySearch.submit();">$linkText</a>
```

5.6 Portlet zur Darstellung und Sortierung einer Tabelle

Dieses Portlet stellt eine Tabelle dar und ermöglicht es, die Tabellenzeilen zu sortieren.

Die Verwendung dieses Portlets ist lizenzierungspflichtig.

5.6.1 Bedienung

Die Tabellenzeilen können nach den Werten in einer Spalte alphabetisch sortiert werden, indem man im Portlet auf den entsprechenden Spaltenkopf klickt. Ein weiterer Klick auf diesen Spaltenkopf kehrt die Sortierrichtung um. Sofern mehr Zeilen vorhanden sind als laut Konfiguration auf einer Seite angezeigt werden dürfen, werden Links zum Blättern angezeigt.

5.6.2 Konfiguration

Das Layout dieses Portlets (wie auch aller anderen) wird mit Hilfe eines Velocity-Templates festgelegt, das in der Web-Applikation `PM-PL` unter `/WEB-INF/templates/com/infopark/portlet/gridview` liegt. Das Layout kann beliebig angepasst werden.

Binden Sie die darzustellende Tabelle als XML-Element in das `npspm`-Element ein. Um Probleme mit Editoren zu vermeiden, sollte ein CDATA-Abschnitt verwendet werden.

5.6.3 Verwendung

Das Portlet kann entsprechend dem folgenden Beispiel in Layoutdateien eingebunden werden:

```
<npspm includePortlet="/PM-PL/gridview" ... ><![CDATA[
  <grid rowsPerPage="10">
    <row><cell>Titel 1</cell><cell>Titel 2</cell></row>
```

```
    <row><cell>1.1</cell><cell>1.2</cell></row>
    <row><cell>2.1</cell><cell>2.2</cell></row>
    <row><cell>3.1</cell><cell>3.2</cell></row>
  </grid>
]]></npspm>
```

Optional können Sie das `rowsPerPage`-Attribut des `grid`-Elements verwenden, um die Anzahl der Zeilen festzulegen, die im Portlet maximal angezeigt werden sollen.

5.7 Benutzerspezifische Portlet-Einstellungen speichern

Portlets können benutzerspezifische Einstellungen (etwa zu Personalisierungszwecken) auf unterschiedliche Art speichern. Der Infopark Portal Manager verfügt über drei Speichervarianten für solche Daten:

- **Im RAM (`MemoryPreferencesStorage`)**: Dies ist die schnellste Methode, jedoch gehen die gespeicherten Informationen verloren, wenn der Application Server neu gestartet wird.
- **Im Dateisystem (`FilesystemPreferencesStorage`)**: Die Einstellungen werden als Java-Preferences im Home-Verzeichnis des Benutzers abgelegt, der den Trifork gestartet hat. Bei vielen Benutzern oder wenn viele Portlets eingesetzt werden, reichen die Speichermöglichkeiten des Dateisystems unter Umständen nicht aus. So kann etwa unter Linux ein Verzeichnis höchstens 32.768 Verzeichnisse enthalten.
- **In einer Datenbank (DatabasePreferencesStorage)** (ab Version 6.7.1): Die Speicherung der Benutzerdaten in einer Datenbank ist die flexibelste Methode, erhöht jedoch den administrativen Aufwand.

Die Speicherart kann im Property `preferencesStorage` mit Hilfe eines Beans in der Datei `pm.xml` der Portlet-Web-Applikation spezifiziert werden. Nur eines dieser Beans darf verwendet werden, deshalb sind im folgenden Beispiel zwei Beans auskommentiert.

```
<property name="preferencesStorage">
  <!-- Store in memory only. Prefs will not survive appserver restart -->
  <!-- bean class="com.infopark.pm.MemoryPreferencesStorage" /> -->
  <bean class="com.infopark.pm.FilesystemPreferencesStorage" />
  <!-- Store in database table. Configure a dataSource bean -->
  <!--
  <bean class="com.infopark.pm.DatabasePreferencesStorage">
    <property name="dataSource" ref="portletPrefsDataSource"/>
    <property name="tableName" value="portlet_prefs"/>
    <property name="portletColumn" value="portlet"/>
    <property name="loginColumn" value="login"/>
    <property name="keyColumn" value="key"/>
    <property name="indexColumn" value="idx"/>
    <property name="valueColumn" value="value"/>
  </bean>
  -->
</property>
```

Die Beans, mit denen die Benutzerdaten im RAM oder im Dateisystem abgelegt werden, haben keine konfigurierbaren Einstellungen.

Das Bean zur Speicherung der Daten in einer Datenbank erwartet, dass die Datenquelle existiert und zugreifbar ist. Die Datenquelle kann im Bean `com.infopark.pm.DatabasePreferencesStorage` mit den folgenden `property`-Elementen konfiguriert werden:

- **`dataSource`**: Referenzieren Sie mit dem Attribut `ref` ein Bean, etwa `portletPrefsDataSource`, mit dem die Klasse für den Zugriff auf die Datenquelle (der Adapter) konfiguriert wird, etwa das folgende Bean für den Zugriff über JDBC:

```
<bean id="portletPrefsDataSource"
 class="org.springframework.jndi.JndiObjectFactoryBean">
  <property name="jndiName">
    <value>jdbc/portletDB</value>
  </property>
</bean>
```

- **`tableName`**: Name der Datenbanktabelle.
- **`indexColumn`**: Name der Spalte, in der der Primärschlüssel abgelegt wird.
- **`keyColumn`**: Name der Spalte, in der der Name des zu speichernden Wertes (der portlet- und benutzerspezifische Parametername) abgelegt wird.
- **`portletColumn`**: Name der Spalte, in der die Portlet-ID gespeichert wird.
- **`loginColumn`**: Name der Spalte, in der der Benutzername gespeichert wird.
- **`valueColumn`**: Name der Spalte, in der der Wert des Parameters abgelegt wird.

Das Datenbank-Interface wird im Infopark Portal Manager ausschließlich vom News-Portlet verwendet. Um das Interface wie im obigen Beispiel zu nutzen, müssen die folgenden Voraussetzungen erfüllt sein:

- **Verfügbare Datenbank**: Die Datenbank muss eingerichtet worden sein und über die oben spezifizierte Tabelle verfügen. Die Werte aller Spalten außer `indexColumn` sind vom Typ `VARCHAR`. Die Werte von `indexColumn` haben den Typ `INT`. Der folgende Datenbankbefehl legt eine solche Tabelle an:

```
CREATE TABLE portlet_prefs (
  idx INT,
  key VARCHAR,
  login VARCHAR,
  portlet VARCHAR,
  value VARCHAR
)
```

- **Installierte JDBC-Treiber**: Der JDBC-Treiber für Ihre Datenbank muss im Trifork-Server installiert worden sein. Sie können ihn über die Console des Servers hochgeladen.
- **Eingerichtete Datenquelle**: Sie benötigen im Trifork-Server eine `dataSource` und eine `pooledDataSource`, um über den JDBC-Treiber auf Ihre eingerichtete Datenbank zuzugreifen.

Sind die Datenbank und der Zugriff auf sie eingerichtet, können Sie in Ihren Portlets analog zum News-Portlet über das API des Beans die benutzerspezifischen Portlet-Einstellungen speichern und auslesen.

5.8 Hinweise zur Entwicklung eigener Portlets

Ab Version 6.0.4 wird der Portal Manager mit einem Java-API geliefert, das zur Entwicklung von Portlets benötigt wird. Das API ist im Verzeichnis `/share/doc/javadoc/portlet-api` zu finden, die Dokumentation der Klassen befindet sich im Verzeichnis `/share/doc/javadoc/pm`. Öffnen Sie die Datei `index.html` in diesem Verzeichnis und beachten Sie bitte auch den Paket-Überblick (*overview*).

Im Installationsverzeichnis befindet sich unterhalb des Verzeichnisses `examples/portlet` eine Portlet-Web-Applikation mit einigen Beispielportlets. Diese kann über das mitgelieferte ant-Buildfile deployed werden und findet sich dann unter `/PM-EX`.

Jede Portlet-Web-Applikation des Infopark Portal Managers enthält ein Servlet, mit dem die Portlets getestet werden können, ohne sie in den Content einzubinden. Dieses Servlet ist unter dem Pfad `/`*`webapp`*`/debug/` zu erreichen, wobei *`webapp`* für die Web-Applikation steht, beispielsweise `/PM-EX/debug/`. Das Servlet gibt die in der Web-Applikation enthaltenen Portlets verlinkt aus, so dass sie durch einfaches Anklicken getestet werden können. Seit Version 6.5.1 kann dieses Servlet auch in der Portal-Manager-Web-Applikation unter `/PM/debug/` aufgerufen werden, um die WSRP-Einbindung zu testen.